

ECAI 2006

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including "Information Modelling and Knowledge Bases" and "Knowledge-Based Intelligent Engineering Systems". It also includes the biannual ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Series Editors:

J. Breuker, R. Dieng, N. Guarino, J.N. Kok, J. Liu, R. López de Mántaras,
R. Mizoguchi, M. Musen and N. Zhong

Volume 141

Recently published in this series

- Vol. 140. E. Tyugu and T. Yamaguchi (Eds.), Knowledge-Based Software Engineering – Proceedings of the Seventh Joint Conference on Knowledge-Based Software Engineering
- Vol. 139. A. Bundy and S. Wilson (Eds.), Rob Milne: A Tribute to a Pioneering AI Scientist, Entrepreneur and Mountaineer
- Vol. 138. Y. Li et al. (Eds.), Advances in Intelligent IT – Active Media Technology 2006
- Vol. 137. P. Hassanaly et al. (Eds.), Cooperative Systems Design – Seamless Integration of Artifacts and Conversations – Enhanced Concepts of Infrastructure for Communication
- Vol. 136. Y. Kiyoki et al. (Eds.), Information Modelling and Knowledge Bases XVII
- Vol. 135. H. Czap et al. (Eds.), Self-Organization and Autonomic Informatics (I)
- Vol. 134. M.-F. Moens and P. Spyns (Eds.), Legal Knowledge and Information Systems – JURIX 2005: The Eighteenth Annual Conference
- Vol. 133. C.-K. Looi et al. (Eds.), Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences – Sharing Good Practices of Research, Experimentation and Innovation
- Vol. 132. K. Nakamatsu and J.M. Abe (Eds.), Advances in Logic Based Intelligent Systems – Selected Papers of LAPTEC 2005
- Vol. 131. B. López et al. (Eds.), Artificial Intelligence Research and Development
- Vol. 130. K. Zieliński and T. Szmac (Eds.), Software Engineering: Evolution and Emerging Technologies
- Vol. 129. H. Fujita and M. Mejri (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the fourth SoMeT_W05
- Vol. 128. J. Zhou et al. (Eds.), Applied Public Key Infrastructure – 4th International Workshop: IWAP 2005
- Vol. 127. P. Ritrovato et al. (Eds.), Towards the Learning Grid – Advances in Human Learning Services
- Vol. 126. J. Cruz, Constraint Reasoning for Differential Models
- Vol. 125. C.-K. Looi et al. (Eds.), Artificial Intelligence in Education – Supporting Learning through Intelligent and Socially Informed Technology
- Vol. 124. T. Washio et al. (Eds.), Advances in Mining Graphs, Trees and Sequences
- Vol. 123. P. Buitelaar et al. (Eds.), Ontology Learning from Text: Methods, Evaluation and Applications
- Vol. 122. C. Mancini, Cinematic Hypertext –Investigating a New Paradigm
- Vol. 121. Y. Kiyoki et al. (Eds.), Information Modelling and Knowledge Bases XVI
- Vol. 120. T.F. Gordon (Ed.), Legal Knowledge and Information Systems – JURIX 2004: The Seventeenth Annual Conference
- Vol. 119. S. Nascimento, Fuzzy Clustering via Proportional Membership Model
- Vol. 118. J. Barzdins and A. Caplinskas (Eds.), Databases and Information Systems – Selected Papers from the Sixth International Baltic Conference DB&IS'2004
- Vol. 117. L. Castillo et al. (Eds.), Planning, Scheduling and Constraint Satisfaction: From Theory to Practice
- Vol. 116. O. Corcho, A Layered Declarative Approach to Ontology Translation with Knowledge Preservation
- Vol. 115. G.E. Phillips-Wren and L.C. Jain (Eds.), Intelligent Decision Support Systems in Agent-Mediated Environments
- Vol. 114. A.C. Varzi and L. Vieu (Eds.), Formal Ontology in Information Systems – Proceedings of the Third International Conference (FOIS-2004)
- Vol. 113. J. Vitrià et al. (Eds.), Recent Advances in Artificial Intelligence Research and Development
- Vol. 112. W. Zhang and V. Sorge (Eds.), Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems

ECAI 2006

17th European Conference on Artificial Intelligence
August 29 – September 1, 2006, Riva del Garda, Italy

Including

Prestigious Applications of Intelligent Systems (PAIS 2006)

Proceedings

Edited by

Gerhard Brewka

Leipzig University, Germany

Silvia Coradeschi

Örebro University, Sweden

Anna Perini

SRA, ITC-irst, Trento, Italy

and

Paolo Traverso

SRA, ITC-irst, Trento, Italy

Organized by the European Coordinating Committee for Artificial Intelligence (ECCAI) and
the Italian Association of Artificial Intelligence

Hosted by the Centro per la Ricerca Scientifica e Technologica (ITC-irst, Trento)

IOS
Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

© 2006 The authors.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without prior written permission from the publisher.

ISBN 1-58603-642-4

Library of Congress Control Number: 2006929617

Publisher

IOS Press

Nieuwe Hemweg 6B

1013 BG Amsterdam

Netherlands

fax: +31 20 687 0019

e-mail: order@iospress.nl

Distributor in the UK and Ireland

Gazelle Books Services Ltd.

White Cross Mills

Hightown

Lancaster LA1 4XS

United Kingdom

fax: +44 1524 63232

e-mail: sales@gazellebooks.co.uk

Distributor in the USA and Canada

IOS Press, Inc.

4502 Rachael Manor Drive

Fairfax, VA 22032

USA

fax: +1 703 323 3668

e-mail: iosbooks@iospress.com

LEGAL NOTICE

The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

ECCAI Member Societies

ACIA (Spain) Catalan Association for Artificial Intelligence (*Associació Catalana d'Inteligència Artificial*)

ADUIS (Ukraine) Association of Developers and Users of Intelligent Systems.

AEPIA (Spain) Spanish Association for Artificial Intelligence (*Asociación Española para la Inteligencia Artificial*)

AFIA (France) French Association for Artificial Intelligence (*Association Française pour l'Intelligence Artificielle*)

AIAI (Ireland) Artificial Intelligence Association of Ireland

AIIA (Italy) Italian Association for Artificial Intelligence (*Associazione Italiana per l'Intelligenza Artificiale*)

AISB (United Kingdom) Society for the Study of Artificial Intelligence and the Simulation of Behaviour

APPIA (Portugal) Portuguese Association for Artificial Intelligence (*Associação Portuguesa para a Inteligência Artificial*)

BAIA (Bulgaria) Bulgarian Artificial Intelligence Association

BCS-SGAI (United Kingdom) British Computer Society Specialist Group on Artificial Intelligence

BNVKI (Belgium/Netherlands) Belgian-Dutch Association for Artificial Intelligence (*Belgisch-Nederlandse Vereniging voor Kunstmatige Intelligentie*)

CSKI (Czech Republic) Czech Society for Cybernetics and Informatics (*Ceská společnost pro kybernetiku a informatiku*)

DAIS (Denmark) Danish Artificial Intelligence Society

EETN (Greece) Hellenic Artificial Intelligence Association

FAIS (Finland) Finnish Artificial Intelligence Society (*Suomen Tekoälyseura ry*)

GI/KI (Germany) German Informatics Association (*Gesellschaft für Informatik; Sektion KI e.V.*)

IAAI (Israel) Israeli Association for Artificial Intelligence

LANO (Latvia) Latvian National Organisation of Automatics (*Latvijas Automatikas Nacionala Organizacija*)

LIKS-AIS (Lithuania) Lithuanian Computer Society–Artificial Intelligence Section (*Lietuvos Kompiuterininkų Sajunga*)

NJSZT (Hungary) John von Neumann Society for Computing Sciences (*Neumann János Számítógéptudományi Társaság*)

ÖGAI (Austria) Austrian Society for Artificial Intelligence (*Österreichische Gesellschaft für Artificial Intelligence*)

RAAI (Russia) Russian Association for Artificial Intelligence

SAIS (Sweden) Swedish Artificial Intelligence Society

SGAICO (Switzerland) Swiss Group for Artificial Intelligence and Cognitive Science (*Schweizer Informatiker Gesellschaft*)

SLAIS (Slovenia) Slovenian Artificial Intelligence Society (*Slovensko društvo za umetno inteligenco*)

SSKI SAV (Slovak Republic) Slovak Society for Cybernetics and Informatics at Slovak Academy of Sciences (*Slovenská spoločnosť pre kybernetiku a informatiku pri Slovenskej akadémii vied*)

This page intentionally left blank

Conference Chair

Silvia Coradeschi, Sweden

Programme Chair

Gerhard Brewka, Germany

Organizing Committee Chairs

Anna Perini, Italy

Paolo Traverso, Italy

Workshop Chair

Toby Walsh, Australia

Advisory Board

Wolfgang Bibel, Germany

Tony Cohn, UK

Werner Horn, Austria

Ramon López de Mántaras, Spain

Lorenza Saitta, Italy

Oliviero Stock, Italy

Wolfgang Wahlster, Germany

Area Chairs

Franz Baader, Germany

Christian Bessiere, France

Susanne Biundo, Germany

Ronen Brafman, Israel

Pádraig Cunningham, Ireland

Barbara Dunin-Keplicz, Poland

Thomas Eiter, Austria

Boi Faltings, Switzerland

Peter Flach, UK

Johannes Fürnkranz, Germany

Hector Geffner, Spain

Malik Ghallab, France

Enrico Giunchiglia, Italy

Lluís Godó, Spain

Vaclav Hlavac, Czech Republic

Sarit Kraus, Israel

Bernhard Nebel, Germany

Ilkka Niemelä, Finland

Francesca Rossi, Italy

Donia Scott, UK

Michèle Sebag, France

Niels Taatgen, Netherlands

Pietro Torasso, Italy

Michael Wooldridge, UK

Poster Chair

Jérôme Lang, France

Poster Programme Committee

Hubie Chen
Eyke Hüllermeier
Frédéric Koriche
Weiru Liu
Nicolas Maudet
Maurice Pagnucco

Helena Sofia Pinto
Pearl Pu
Alessandro Saffiotti
Alessandro Sperduti
Leendert van der Torre
Laure Vieu

PAIS Chair

Mugur Tatar, Germany

PAIS Programme Committee

Josep Lluis Arcos
Joachim Baumeister
Riccardo Bellazzi
Bertrand Braunschweig
Pádraig Cunningham
Floriana Esposito

Boi Faltings
Gerhard Fleischanderl
Oliviero Stock
Kilian Stoffel
Louise Trave-Massuyes
Franz Wotawa

ECAI Programme Committee

Eyal Amir	Frank Dignum	Jörg Hoffmann
Elisabeth André	Yannis Dimopoulos	David Hogg
Liliana Ardissono	Jürgen Dix	Andreas Hotho
Ronald Arkin	Dmitri Dolgov	Andrew Howes
Paolo Avesani	Carmel Domshlak	Eyke Hüllermeier
Ruth Aylett	Alexis Drogoul	Anthony Hunter
Chitta Baral	Paul Dunne	Giovambattista Ianni
Roman Bartak	Stefan Edelkamp	Félix Ingrand
Mathias Bauer	Jan-Olof Eklundh	Wojtek Jamroga
Peter Baumgartner	Amal El Fallah Seghrouchni	Tomi Janhunen
Michael Beetz	Tapio Elooma	Nicholas Jennings
Trevor Bench-Capon	Ulle Endriss	Peter Jonsson
Rachel Ben-Eliyahu-Zohary	Esra Erdem	Ulrich Junker
Salem Benferhat	Dominique Estival	Narendra Jussien
Brandon Bennett	Wolfgang Faber	Antonis Kakas
Michael Berthold	Rino Falcone	Gabriele Kern-Isberner
Stefano Bistarelli	Ad Feelders	Kristian Kersting
Hendrik Blockeel	Michael Felsberg	Marek Kisiel-Dorohinicki
Nadia Bolshakova	Michael Fink	Zeynep Kiziltan
Blai Bonet	Filippo Focacci	Reinhard Klette
Rafael Bordini	Norman Foo	Jana Koehler
Daniel Borrajo	Wolfgang Förstner	Sven Koenig
Henrik Boström	Maria Fox	Michael Kohlhase
Pavel Brazdil	Christian Freksa	Joost Kok
Derek Bridge	Alan Frisch	Jana Kosecka
Francesco Buccafurri	Uli Furbach	Manolis Koubarakis
Marco Cadoli	Alfredo Gabaldon	Tim Kovacs
Diego Calvanese	João Gama	Stefan Kramer
Stéphane Canu	Michael Gelfond	Philippe Laborie
John Carroll	Claudio Gentile	Nicolas Lachiche
Luis Castillo	Laura Giordano	Gerhard Lakemeyer
Tristan Cazenave	Marie-Pierre Gleizes	Mirella Lapata
Jesús Cerquides	Rajeev Goré	Pedro Larrañaga
Amedeo Cesta	Laurent Granvilliers	Javier Larrosa Bondia
Krzysztof Cetnarowicz	Gianluigi Greco	Christian Laugier
Alessandro Cimatti	Gunter Grieser	Jean-Paul Laumond
Stephen Clark	Michal Haindl	Daniel Le Berre
Marie-Odile Cordier	Eric Hansen	Christophe Lecoutre
Antoine Cornuéjols	Patrik Haslum	João Leite
Anna Cox	Salima Hassas	John Levine
Michel Crucianu	Malte Helmert	Paolo Liberatore
Carlos Damásio	Laurent Henocque	Alessio Lomuscio
Luc De Raedt	José Hernández-Orallo	Derek Long
Anne De Roeck	Joachim Hertzberg	Thomas Lukasiewicz
Marina De Vos	Andreas Herzig	Carsten Lutz
Thierry Declerck	Ray Hickey	Michael Madden
James Delgrande	Brahim Hnich	Donato Malerba
Marc Denecker	Jerry Hobbs	Shaul Markovitch

João Marques-Silva	Jochen Renz	Michael Thielscher
Pierre Marquis	Martin Riedmiller	Cesare Tinelli
Stan Matwin	Jussi Rintanen	Hans Tompits
Jasna Maver	Riccardo Rosati	Paolo Torroni
Lee McCluskey	Jeffrey Rosenschein	Paolo Traverso
Amnon Meisels	Michel Rueher	Emanuele Trucco
Chris Mellish	Wheeler Ruml	Mirek Truszczyński
Pedro Meseguer	Régis Sabbadin	Alexey Tsymbal
Nicolas Meuleau	Alessandro Saffiotti	Hudson Turner
John-Jules Meyer	Gerhard Sagerer	Leon Urbas
Ian Miguel	Lorenza Saitta	Enric Vallduví
Michela Milano	Chiaki Sakama	Peter van Beek
Ralf Möller	Ulrike Sattler	Wiebe van der Hoek
Serafín Moral	Francesco Scarcello	Rogier van Eijk
Katharina Morik	Torsten Schaub	Gertjan van Noord
Benoit Morisset	Tobias Scheffer	Hedderik van Rijn
Bertrand Neveu	Thomas Schiex	Maarten van Someren
Thomas Nielsen	Wolfgang Schoppek	Kristen Brent Venable
Wim Nuijten	Marc Sebban	Rineke Verbrugge
Eva Onaindia	Bart Selman	Felisa Verdejo
Barry O'Sullivan	Guy Shani	Gérard Verfaillie
Luigi Palopoli	Carles Sierra	Dirk Vermeir
David Patterson	Reid Simmons	José Vidal
David Pearce	Laurent Simon	Vincent Vidal
Michał Pechoucek	John Slaney	Richard Wallace
David Peebles	Tran Cao Son	Dieter Wallach
Wojciech Penczek	Steffen Staab	Toby Walsh
Laurent Perron	Konstantinos Stergiou	Kewen Wang
Thierry Petit	Andrea Stocco	Gerhard Widmer
Bernhard Pfahringer	Umberto Straccia	Marco Wiering
Gerald Pfeifer	Peter Struss	Mary-Anne Williams
Fabio Pianesi	Milan Studený	Nic Wilson
Jeremy Pitt	Markus Stumptner	Frank Wolter
Paul Piwek	Einoshin Suzuki	Stefan Woltran
Henry Prakken	Tomas Svoboda	Stefan Wölfl
Steve Prestwich	Tamas Szirányi	Franz Wotawa
Patrick Prosser	Armando Tacchella	Stefan Wrobel
Gregory Provan	Daniele Theseider Dupré	Osher Yadgar
Ioannis Refanidis	Sylvie Thiébaut	Jia-Huai You

Additional Reviewers

Thomas Agotnes
 Manuel Alcántara Plá
 Vincent Aleven
 Christian Anger
 Alessandro Artale
 Nathalie Aussenac
 Vincent Barichard
 John Bateman
 Petr Benda
 Piergiorgio Bertoli
 David Billington
 Guido Boella
 Sebastian Brandt
 Agnès Braud
 Michael Brenner
 Wolfram Burgard
 Francesco Calmieri
 Hadrien Cambazard
 Valérie Camps
 Davy Capera
 Andre Carvalho
 Allen Chang
 Carlos Chesñevar
 Yann Chevaleyre
 Daoud Clarke
 Jens Claßen
 Ann Copestake
 Oscar Corcho
 Fabrizio Costa
 Victor de Boer
 Paul de Clercq
 Simon de Givry
 Virginia Dignum
 Conrad Drescher
 Kurt Driessens
 Yagil Engel
 Sofia Espinosa
 Hélène Fargier
 Cèsar Ferri
 Lukas Folty
 Elisa Fromont
 Wolfgang Gaissmaier
 Ignazio Gallo
 Christophe Garcia
 Martin Gebser
 Jean-Pierre Georgé
 Chiara Ghidini
 Valentina Ghiozzi

Birte Glimm
 Claudia Goldman-Shenhar
 Bernardo Cuenca Grau
 Emmanuel Guéré
 Jérémie Guichet
 Fabian Güiza
 Hannaneh Hajishirzi
 Katsutoshi Hirayama
 Pascal Hitzler
 Jan Hladik
 Jiri Hodík
 Michael Hofbaur
 Samuel Jeong
 Liliana Ironi
 Gizela Jakubowska
 Yi Jin
 Pavel Jisl
 Denis Jouvin
 Souhila Kaci
 Magda Kacprzak
 Alissa Kaplunova
 Atila Kaya
 Manfred Kerber
 Alexander Kleiner
 Rob Koeling
 Kathrin Konczak
 Barteld Kooi
 Tomasz Kowalski
 Oliver Kutz
 Kate Larson
 Domenico Lembo
 Leonardo Lesmo
 Guohua Liu
 Xavier Lorca
 Peter Lucas
 Marco Maratea
 Alberto Martelli
 Yves Martin
 Nicolas Maudet
 Thomas Meyer
 Pavlos Moraitis
 Boris Motik
 Marie-Laure Mugnier
 Artur Niewiadomski
 Peter Novak
 Oliver Obst
 Juan Fernández Olivares
 Manuel Gómez Olmeto

Alessandro Oltramari
 Eric Pacuit
 Praveen Paruchuri
 Federico Pecora
 Patrice Perny
 Simona Perri
 Jochen Pfalzgraf
 Helena Sofia Pinto
 Marco Pistore
 Andrei Popescu
 Gianluca Pozzato
 Cédric Pralet
 Laurent Prevot
 Josep Puyol-Gruart
 Frédéric Py
 Guilin Qi
 Silvana Quaglini
 René Quiniou
 Stefan Raeymaekers
 Marco Ragni
 Deepak Ramachandran
 Oliver Ray
 Martin Rehak
 Francesco Ricca
 Pedro Rodrigues
 Milan Rollo
 Niall Rooney
 Grigore Rosu
 Olivier Roy
 Sandra Sandri
 Sebastian Sardinia
 Leander Schietgat
 Roman Schindlauer
 Stefan Schlobach
 Meinolf Sellmann
 Luciano Serafini
 Mathieu Serrurier
 Dafna Shahaf
 Steven Shapiro
 Afsaneh Shirazi
 Guillermo Simari
 Barry Smyth
 Viorica Sofronie-Stokkermans
 Gerd Stumme
 Maciej Szreter
 Wouter Teepe
 Paolo Terenziani
 Elias Thijssse

Vicenç Torra
Octavian Udrea
Werner Uwents
Willem-Jan van Hoeve
Joaquin Vanschoren
Bart Verheij

Thuc Vu
Dirk Walther
Michael Wessel
Cees Witteveen
Makoto Yokoo
Michael Zakharyashev

Dongmo Zhang
Yan Zhang
Yingqian Zhang
Yuting Zhao
Vittorio Ziparo
Jell Zuidema

Preface

In the summer of 1956, John McCarthy organized the famous Dartmouth Conference which is now commonly viewed as the founding event for the field of Artificial Intelligence. During the last 50 years, AI has seen a tremendous development and is now a well-established scientific discipline all over the world. Also in Europe AI is in excellent shape, as witnessed by the large number of high quality submissions we received. About 600 papers and posters were registered for ECAI-06, out of which 550 were actually reviewed (501 paper submissions and 49 poster submissions). The program committee decided to accept 131 full papers, which amounts to an acceptance rate of 26.1%, and 75 posters (authors of full papers had the possibility to opt for acceptance as poster in case of rejectance as full paper).

We received submissions from 43 different countries, and accepted papers from 25 countries. The following table shows the number of submissions and accepted papers per country, based on the contact author affiliation:

Algeria	5	0	Greece	15	4	Romania	4	1
Australia	19	9	Hungary	1	0	Russia	3	0
Austria	9	5	India	1	0	Singapore	4	4
Belgium	4	2	Iran	5	1	Spain	32	5
Brazil	12	0	Ireland	14	9	Slovenia	2	2
Bulgaria	1	0	Israel	8	2	Slovakia	3	3
Canada	13	4	Italy	83	36	Sweden	5	5
China	4	0	Japan	9	1	Switzerland	5	3
Cyprus	1	0	Luxemburg	5	3	Thailand	2	0
Czechia	4	0	Mexico	2	0	Turkey	3	0
Egypt	1	0	Netherlands	26	12	UK	49	22
Estonia	2	0	New Zealand	3	0	USA	22	5
France	111	46	Pakistan	1	0	Venezuela	1	1
Finland	3	2	Poland	3	0			
Germany	47	19	Portugal	1	0			

It is also interesting to look at the areas of the submitted and accepted papers/posters. We show both absolute numbers and percentage. The area information is based on the first two key words chosen by the authors:

	# submitted	%	# accepted	%
Case-Based Reasoning	10.5	1.9	0.5	0.2
Cognitive Modelling	33.5	6.1	13	6.3
Constraints & Search	54.5	10.0	26	12.6
Distributed AI/Agents	107	19.6	36.5	17.7
KR & Reasoning	141.5	25.9	55.5	26.9
Machine Learning	83	15.2	29	14.1
Model-Based Reasoning	32	5.9	8	3.9
Natural Language	21.5	3.9	7.5	3.6
Perception/Vision	6	1.1	2	1.0
Planning and Scheduling	27	4.9	12	5.8
Robotics	12.5	2.3	5	2.4
PAIS	18	3.3	11	5.3

In comparison with ECAI 2004, we see a strong increase in the relative number of submissions from Distributed AI/Agents and Cognitive Modelling. Knowledge Representation & Reasoning is traditionally strong in Europe and remains the biggest area of ECAI-06. One reason the figures for Case-Based Reasoning are rather low is that much of the high quality work in this area has found its way into prestigious applications and is thus represented under the heading of PAIS.

The ECAI-06 best paper award, sponsored by Elsevier, goes to a machine learning paper:

A Real Generalization of Discrete AdaBoost, by Richard Nock and Frank Nielsen

Congratulations! The best poster award, sponsored by IOS Press, will be decided after the poster sessions in Riva. The 10 best papers are invited to a fast track of the Artificial Intelligence Journal.

A conference of the size of ECAI needs a lot of support from many people. At this point we would like to thank all those who helped to make ECAI-06 a tremendous success: the poster, workshop, PAIS and area chairs faced a heavy workload and they all did an excellent job. The PC members and additional reviewers came up with timely, high quality reviews and made the life of the PC chair as easy as it can be. Thanks also to Alex Nittka who ran the conference management software, to the PC chair's great relief. We also want to thank the many people involved in the local organization of the conference: there would be no conference without you.

Our very special thanks go to a person who is no longer with us. Rob Milne, chair of ECAI-06 until he died of a heart attack close to the summit of Mount Everest on June 5th, 2005. He shaped this conference from the beginning, and we did our best to organize ECAI-06 in his spirit.

June 2006

Gerhard Brewka
Silvia Coradeschi
Anna Perini
Paolo Traverso

Contents

ECCAI Member Societies	v
Conference Organization	vii
ECAI Programme Committee	ix
Additional Reviewers	xi
Preface	xiii

Gerhard Brewka, Silvia Coradeschi, Anna Perini and Paolo Traverso

I. Invited Talks

Socially Intelligent Robots <i>Cynthia Breazeal</i>	3
Managing Diversity in Knowledge <i>Fausto Giunchiglia</i>	4
The Truth About Defaults <i>Hector Levesque</i>	5
SmartWeb: Getting Answers on the Go <i>Wolfgang Wahlster</i>	6

II. Papers

1. Cognitive Modelling

Background Default Knowledge and Causality Ascriptions <i>Jean-François Bonnefon, Rui Da Silva Neves, Didier Dubois and Henri Prade</i>	11
Comparing Sets of Positive and Negative Arguments: Empirical Assessment of Seven Qualitative Rules <i>Jean-François Bonnefon and Hélène Fargier</i>	16
A Similarity and Fuzzy Logic-Based Approach to Cerebral Categorisation <i>Julien Erny, Josette Pastor and Henri Prade</i>	21
Imitation of Intentional Behaviour <i>Bart Jansen</i>	26
Dramatization Meets Narrative Presentations <i>Rossana Damiano, Vincenzo Lombardo, Antonio Pizzo and Fabrizio Nunnari</i>	31
MAMA: An Architecture for Interactive Musical Agents <i>David Murray-Rust, Alan Smaill and Michael Edwards</i>	36
Bayesian Modelling of Colour's Usage Impact to Web Credibility <i>Eleftherios Papachristos, Nikolaos Tselios and Nikolaos Avouris</i>	41
Evaluating Perception of Interaction Initiation in Virtual Environments Using Humanoid Agents <i>Christopher Peters</i>	46
Cognitive Situated Agents Learn to Name Actions <i>Julien Poudade, Lionel Landwerlin and Patrick Paroubek</i>	51

Tracking the Lexical *Zeitgeist* with WordNet and Wikipedia*Tony Veale*

56

2. Constraints and Search

Enhancing Constraints Manipulation in Semiring-Based Formalisms <i>Stefano Bistarelli and Fabio Gadducci</i>	63
Evaluating ASP and Commercial Solvers on the CSPLib <i>Marco Cadoli, Toni Mancini, Davide Micaletto and Fabio Patrizi</i>	68
Automatic Generation of Implied Constraints <i>John Charnley, Simon Colton and Ian Miguel</i>	73
Maintaining Generalized Arc Consistency on Ad-Hoc n -Ary Boolean Constraints <i>Kenil C.K. Cheng and Roland H.C. Yap</i>	78
A Study on the Short-Term Prohibition Mechanisms in Tabu Search <i>Luca Di Gaspero, Marco Chiarandini and Andrea Schaerf</i>	83
Random Subset Optimization <i>Boi Faltings and Quang-Huy Nguyen</i>	88
Search for Compromise Solutions in Multiobjective State Space Graphs <i>Lucie Galand and Patrice Perny</i>	93
MINION: A Fast Scalable Constraint Solver <i>Ian P. Gent, Chris Jefferson and Ian Miguel</i>	98
Asynchronous Forward-Bounding for Distributed Constraints Optimization <i>Amir Gershman, Amnon Meisels and Roie Zivan</i>	103
Distributed Log-Based Reconciliation <i>Yek Loong Chong and Youssef Hamadi</i>	108
Extracting MUCs from Constraint Networks <i>Fred Hemery, Christophe Lecoutre, Lakhdar Sais and Frédéric Boussemart</i>	113
Preference-Based Inconsistency Proving: When the Failure of the Best Is Sufficient <i>Ulrich Junker</i>	118
Return of the JTMS: Preferences Orchestrate Conflict Learning and Solution Synthesis <i>Ulrich Junker and Olivier Lhomme</i>	123
Multi-Objective Propagation in Constraint Programming <i>Emma Rollon and Javier Larrosa</i>	128
Last Conflict Based Reasoning <i>Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary and Vincent Vidal</i>	133
Dynamic Orderings for AND/OR Branch-and-Bound Search in Graphical Models <i>Radu Marinescu and Rina Dechter</i>	138
Compact Representation of Sets of Binary Constraints <i>Jussi Rintanen</i>	143
Pessimistic Heuristics Beat Optimistic Ones in Real-Time Search <i>Aleksander Sadikov and Ivan Bratko</i>	148

Inverse Consistencies for Non-Binary Constraints <i>Kostas Stergiou and Toby Walsh</i>	153
Guiding Search Using Constraint-Level Advice <i>Radoslaw Szymanek and Barry O'Sullivan</i>	158
Beyond Singleton Arc Consistency <i>Marc van Dongen</i>	163
Symmetry Breaking Using Value Precedence <i>Toby Walsh</i>	168
3. Distributed AI/Agents	
Coordination Through Inductive Meaning Negotiation <i>Alessandro Agostini</i>	175
Reaching Agreements for Coalition Formation Through Derivation of Agents' Intentions <i>Samir Aknine and Onn Shehory</i>	180
Cheating Is Not Playing: <i>Methodological Issues of Computational Game Theory</i> <i>Bruno Beaujalis and Philippe Mathieu</i>	185
Mediation in the Framework of Morpho-Logic <i>Isabelle Bloch, Ramón Pino-Pérez and Carlos Uzcátegui</i>	190
Strengthening Admissible Coalitions <i>Guido Boella, Luigi Sauro and Leendert van der Torre</i>	195
Advanced Policy Explanations on the Web <i>Piero Bonatti, Daniel Olmedilla and Joachim Peer</i>	200
Prevention of Harmful Behaviors Within Cognitive and Autonomous Agents <i>Caroline Chopinaud, Amal El Fallah Seghrouchni and Patrick Taillibert</i>	205
Coalition Structure Generation in Task-Based Settings <i>Viet Dung Dang and Nicholas R. Jennings</i>	210
Programming Agents with Emotions <i>Mehdi Dastani and John-Jules Ch. Meyer</i>	215
Goal Types in Agent Programming <i>Mehdi Dastani, M. Birna van Riemsdijk and John-Jules Ch. Meyer</i>	220
Alternating-Offers Bargaining Under One-Sided Uncertainty on Deadlines <i>Francesco Di Giunta and Nicola Gatti</i>	225
A Logic-Based Framework to Compute Pareto Agreements in One-Shot Bilateral Negotiation <i>Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio and Francesco M. Donini</i>	230
Towards ACL Semantics Based on Commitments and Penalties <i>Leila Amgoud and Florence Dupin de Saint-Cyr</i>	235
Computational Opinions <i>Felix Fischer and Matthias Nickles</i>	240
A New Semantics for the FIPA Agent Communication Language Based on Social Attitudes <i>Benoit Gaudou, Andreas Herzig, Dominique Longin and Matthias Nickles</i>	245

Contouring of Knowledge for Intelligent Searching for Arguments <i>Anthony Hunter</i>	250
On the Inability of Gathering by Asynchronous Mobile Robots with Initial Movements <i>Branislav Katreňák and Jana Katreňáková</i>	255
Testing the Limits of Emergent Behavior in MAS Using Learning of Cooperative Behavior <i>Jordan Kidney and Jörg Denzinger</i>	260
Boolean Games Revisited <i>Elise Bonzon, Marie-Christine Lagasquie-Schiex, Jérôme Lang and Bruno Zanuttini</i>	265
An Automated Agent for Bilateral Negotiation with Bounded Rational Agents with Incomplete Information <i>Raz Lin, Sarit Kraus, Jonathan Wilkenfeld and James Barry</i>	270
Self-Organizing Multiagent Approach to Optimization in Positioning Problems <i>Sana Moujahed, Olivier Simonin, Abderrafiâ Koukam and Khaled Ghédira</i>	275
Arguing with Confidential Information <i>Nir Oren, Timothy J. Norman and Alun Preece</i>	280
Auction Mechanisms for Efficient Advertisement Selection on Public Displays <i>Terry Payne, Ester David, Nicholas R. Jennings and Matthew Sharifi</i>	285
Verifying Interlevel Relations Within Multi-Agent Systems <i>Alexei Sharpanskykh and Jan Treur</i>	290
Flexible Provisioning of Service Workflows <i>Sebastian Stein, Nicholas R. Jennings and Terry R. Payne</i>	295
Heuristic Bidding Strategies for Multiple Heterogeneous Auctions <i>David C.K. Yuen, Andrew Byde and Nicholas R. Jennings</i>	300
Are Parallel BDI Agents Really Better? <i>Huiliang Zhang and Shell Ying Huang</i>	305
Dynamic Control of Intention Priorities of Human-Like Agents <i>Huiliang Zhang and Shell Ying Huang</i>	310

4. Knowledge Representation and Reasoning

Knowing Minimum/Maximum n Formulae <i>Thomas Ågotnes and Natasha Alechina</i>	317
Modal Logics for Communicating Rule-Based Agents <i>Natasha Alechina, Mark Jago and Brian Logan</i>	322
Causation as Production <i>John Bell</i>	327
Merging Possibilistic Networks <i>Salem Benferhat</i>	332
Compiling Possibilistic Knowledge Bases <i>Salem Benferhat and Henri Prade</i>	337
Improving Bound Propagation <i>Bozhena Bidyuk and Rina Dechter</i>	342

Logic Programs with Multiple Chances <i>Francesco Buccafurri, Gianluca Caminiti and Domenico Rosaci</i>	347
Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems <i>Philippe Chatalic, Gia Hien Nguyen and Marie-Christine Rousset</i>	352
Conceptual Hierarchies Matching: An Approach Based on Discovery of Implication Rules Between Concepts <i>Jérôme David, Fabrice Guillet, Régis Gras and Henri Briand</i>	357
CTL Model Update: Semantics, Computations and Implementation <i>Yulin Ding and Yan Zhang</i>	362
Resolving Conflicts in Action Descriptions <i>Thomas Eiter, Esra Erdem, Michael Fink and Ján Senko</i>	367
Possibilistic Influence Diagrams <i>Laurent Garcia and Régis Sabbadin</i>	372
Solving Optimization Problems with DLL <i>Enrico Giunchiglia and Marco Maratea</i>	377
Discovering Missing Background Knowledge in Ontology Matching <i>Fausto Giunchiglia, Pavel Shvaiko and Mikalai Yatskevich</i>	382
Extracting MUSes <i>Éric Grégoire, Bertrand Mazure and Cédric Piette</i>	387
On Probing and Multi-Threading in PLATYPUS <i>Jean Gressmann, Tomi Janhunen, Robert Mercer, Torsten Schaub, Sven Thiele and Richard Tichy</i>	392
Elaborating Domain Descriptions <i>Andreas Herzig, Laurent Perrussel and Ivan Varzinczak</i>	397
On the Logic of Theory Change: Relations Between Incision and Selection Functions <i>Marcelo A. Falappa, Eduardo L. Fermé and Gabriele Kern-Isberner</i>	402
Representing Relative Direction as a Binary Relation of Oriented Points <i>Reinhard Moratz</i>	407
Modular Equivalence for Normal Logic Programs <i>Emilia Oikarinen and Tomi Janhunen</i>	412
Preference Representation with 3-Points Intervals <i>Meltem Öztürk and Alexis Tsoukiàs</i>	417
Reference-Dependent Qualitative Models for Decision Making Under Uncertainty <i>Patrice Perny and Antoine Rolland</i>	422
Decision with Uncertainties, Feasibilities, and Utilities: Towards a Unified Algebraic Framework <i>Cédric Pralet, Gérard Verfaillie and Thomas Schiex</i>	427
Using Occlusion Calculi to Interpret Digital Images <i>David Randell and Mark Witkowski</i>	432
Abductive Logic Programming in the Clinical Management of HIV/AIDS <i>Oliver Ray, Athos Antoniades, Antonis Kakas and Ioannis Demetriadis</i>	437
Interleaving Belief Updating and Reasoning in Abductive Logic Programming <i>Fariba Sadri and Francesca Toni</i>	442

Bridging the Gap Between Informal and Formal Guideline Representations <i>Andreas Seyfang, Silvia Miksch, Mar Marcos, Jolanda Wittenberg, Cristina Polo-Conde and Kitty Rosenbrand</i>	447
Boolean Propagation Based on Literals for Quantified Boolean Formulae <i>Igor Stéphan</i>	452
General Concept Inclusions in Fuzzy Description Logics <i>Giorgos Stoilos, Umberto Straccia, Giorgos Stamou and Jeff Z. Pan</i>	457
Approximating Extended Answer Sets <i>Davy Van Nieuwenborgh, Stijn Heymans and Dirk Vermeir</i>	462
An Axiomatic Approach in Qualitative Decision Theory with Binary Possibilistic Utility <i>Paul Weng</i>	467
An Efficient Upper Approximation for Conditional Preference <i>Nic Wilson</i>	472
A Solver for QBFs in Nonprenex Form <i>Uwe Egly, Martina Seidl and Stefan Woltran</i>	477
Knowledge Engineering for Bayesian Networks: How Common Are Noisy-MAX Distributions in Practice? <i>Adam Zagorecki and Marek Druzdzel</i>	482
5. Machine Learning	
A Unified Model for Multilabel Classification and Ranking <i>Klaus Brinker, Johannes Fürnkranz and Eyke Hüllermeier</i>	489
Learning by Automatic Option Discovery from Conditionally Terminating Sequences <i>Sertan Girgin, Faruk Polat and Reda Alhajj</i>	494
Least Squares SVM for Least Squares TD Learning <i>Tobias Jung and Daniel Polani</i>	499
Argument Based Rule Learning <i>Martin Možina, Jure Žabkar and Ivan Bratko</i>	504
A Real Generalization of Discrete AdaBoost <i>Richard Nock and Frank Nielsen</i>	509
Discovery of Entailment Relations from Event Co-Occurrences <i>Viktor Pekar</i>	516
Efficient Knowledge Acquisition for Extracting Temporal Relations <i>Son Bao Pham and Achim Hoffmann</i>	521
Efficient Learning from Massive Spatial-Temporal Data Through Selective Support Vector Propagation <i>Yilian Qin and Zoran Obradovic</i>	526
Automatic Term Categorization by Extracting Knowledge from the Web <i>Leonardo Rigutini, Ernesto Di Iorio, Marco Ernandes and Marco Maggini</i>	531
Strategic Foresighted Learning in Competitive Multi-Agent Games <i>Pieter Jan 't Hoen, Sander Bohte and Han La Poutré</i>	536

6. Natural Language Processing

MSDA: Wordsense Discrimination Using Context Vectors and Attributes <i>Abdulrahman Almuhareb and Massimo Poesio</i>	543
Integrating Domain and Paradigmatic Similarity for Unsupervised Sense Tagging <i>Roberto Basili, Marco Cammisa and Alfio Massimiliano Gliozzo</i>	548
Disambiguating Personal Names on the Web Using Automatically Extracted Key Phrases <i>Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka</i>	553
History-Based Inside-Outside Algorithm <i>Heshaam Feili and Gholamreza Ghassem-Sani</i>	558
Shallow Semantic Parsing Based on FrameNet, VerbNet and PropBank <i>Ana-Maria Giuglea and Alessandro Moschitti</i>	563
Semantic Tree Kernels to Classify Predicate Argument Structures <i>Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin and Roberto Basili</i>	568

7. Planning and Scheduling

A Multivalued Logic Model of Planning <i>Marco Baioletti, Alfredo Milani, Valentina Poggioni and Silvia Suriani</i>	575
Strong Cyclic Planning Under Partial Observability <i>Piergiorgio Bertoli, Alessandro Cimatti and Marco Pistore</i>	580
Approximation Properties of Planning Benchmarks <i>Malte Helmert, Robert Mattmüller and Gabi Röger</i>	585
Approximate Linear-Programming Algorithms for Graph-Based Markov Decision Processes <i>Nicklas Forsell and Régis Sabbadin</i>	590
Mean Field Approximation of the Policy Iteration Algorithm for Graph-Based Markov Decision Processes <i>Nathalie Peyrard and Régis Sabbadin</i>	595
Unified Definition of Heuristics for Classical Planning <i>Jussi Rintanen</i>	600

8. PAIS

Applying Trip@dvce Recommendation Technology to www.visiteurope.com <i>Adriano Venturini and Francesco Ricci</i>	607
Natural and Intuitive Multimodal Dialogue for In-Car Applications: The SAMMIE System <i>Tilman Becker, Nate Blaylock, Ciprian Gerstenberger, Ivana Kruijff-Korbayová, Andreas Korthauer, Manfred Pinkal, Michael Pitz, Peter Poller and Jan Schehl</i>	612
A Client/Server User-Based Collaborative Filtering Algorithm: Model and Implementation <i>Sylvain Castagnos and Anne Boyer</i>	617
Software Companion: The MEXAR2 Support to Space Mission Planners <i>Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, Angelo Oddi and Nicola Policella</i>	622
ECUE: A Spam Filter that Uses Machine Learning to Track Concept Drift <i>Sarah Jane Delany, Pádraig Cunningham and Barry Smyth</i>	627

Knowledge-Based Recommendation: Technologies and Experiences from Projects <i>Alexander Felfernig, Klaus Isak and Christian Russ</i>	632
Diagnosing Highly Configurable Products: Troubleshooting Support for Airbus Final Assembly Line <i>Andreas Junghanns and Mugur Tatar</i>	637
Web-Based Tools for Codification with Medical Ontologies in Switzerland <i>Thorsten Kurz and Kilian Stoffel</i>	642
Model-Based Failure Analysis with RODON <i>Karin Lunde, Rüdiger Lunde and Burkhard Münker</i>	647
9. Perception	
A Learning Classifier Approach to Tomography <i>Kees Joost Batenburg</i>	655
Depth Ordering and Figure-Ground Segregation in Monocular Images Derived from Illusory Contour Perception <i>Marcus Hund and Bärbel Mertsching</i>	660
Graph Neural Networks for Object Localization <i>Gabriele Monfardini, Vincenzo Di Massa, Franco Scarselli and Marco Gori</i>	665
10. Robotics	
Situation Assessment for Sensor-Based Recovery Planning <i>Abdelbaki Bouguerra, Lars Karlsson and Alessandro Saffiotti</i>	673
Learning Behaviors Models for Robot Execution Control <i>Guillaume Infantes, Félix Ingrand and Malik Ghallab</i>	678
Plan-Based Configuration of a Group of Robots <i>Robert Lundh, Lars Karlsson and Alessandro Saffiotti</i>	683
III. Posters	
1. Cognitive Modeling	
Agents with Anticipatory Behaviors: To Be Cautious in a Risky Environment <i>Cristiano Castelfranchi, Rino Falcone and Michele Piunti</i>	693
AI and Music: Toward a Taxonomy of Problem Classes <i>Oliver Kramer, Benno Stein and Jürgen Wall</i>	695
Using Emotions for Behaviour-Selection Learning <i>Maria Malfaz and Miguel A. Salichs</i>	697
2. Constraints and Search	
Efficient Handling of Complex Local Problems in Distributed Constraint Optimization <i>David A. Burke and Kenneth N. Brown</i>	701
Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms <i>Steven Halim, Roland H.C. Yap and Hoong Chuin Lau</i>	703

Bipolar Preference Problems <i>Stefano Bistarelli, Maria Silvia Pini, Francesca Rossi and K. Brent Venable</i>	705
A Connectivity Constraint Using Bridges <i>Patrick Prosser and Chris Unsworth</i>	707
3. Distributed AI/Agents	
Search Better and Gain More: Investigating New Graph Structures for Multi-Agent Negotiations <i>Samir Aknine</i>	711
An Argumentation-Based Framework for Designing Dialogue Strategies <i>Leila Amgoud and Nabil Hameurlain</i>	713
A Multiagent System for Scheduling Activities Onboard a Space System <i>Francesco Amigoni, Simone Farè, Michèle Lavagna and Guido Sangiovanni</i>	715
Benefits of Combinatorial Auctions with Transformability Relationships <i>Andrea Giovannucci, Jesús Cerquides and Juan Antonio Rodríguez-Aguilar</i>	717
Count-As Conditionals, Classification and Context <i>Guido Boella and Leendert van der Torre</i>	719
Fair Distribution of Collective Obligations <i>Guido Boella and Leendert van der Torre</i>	721
Proactive Identification of a Physician's Information Needs <i>Loes M.M. Braun, Floris Wiesman, H. Jaap van den Herik and Arie Hasman</i>	723
Acyclic Argumentation: Attack = Conflict + Preference <i>Souhila Kaci, Leendert van der Torre and Emil Weydert</i>	725
Semantic Knowledge Model and Architecture for Agents in Discrete Environments <i>Michal Laclavík, Marian Babík, Zoltan Balogh, Emil Gatial and Ladislav Hluchý</i>	727
Partial Local FriendQ Multiagent Learning: Application to Team Automobile Coordination Problem <i>Julien Laumonier and Brahim Chaib-draa</i>	729
Mutual Enrichment for Agents Through Nested Belief Change: A Semantic Approach <i>Laurent Perrussel, Jean-Marc Thévenin and Thomas Meyer</i>	731
Multi-Agent Least-Squares Policy Iteration <i>Victor Palmer</i>	733
4. Knowledge Representation and Reasoning	
Finding Instances of Deduction and Abduction in Clinical Experimental Transcripts <i>Maria Amalfi, Katia Lo Presti, Alessandro Provetti and Franco Salvetti</i>	737
A Semantics for Active Logic <i>Mikael Asker and Jacek Malec</i>	739
An Alternative Inference for Qualitative Choice Logic <i>Salem Benferhat, Daniel Le Berre and Karima Sedki</i>	741
On the Existence of Answer Sets in Normal Extended Logic Programs <i>Martin Caminada and Chiaki Sakama</i>	743

Smoothed Particle Filtering for Dynamic Bayesian Networks <i>Theodore Charitos</i>	745
Goal Revision for a Rational Agent <i>Célia da Costa Pereira, Andrea G.B. Tettamanzi and Leila Amgoud</i>	747
A Redundancy-Based Method for Relation Instantiation from the Web <i>Viktor de Boer, Maarten van Someren and Bob J. Wielinga</i>	749
Norms with Deadlines in Dynamic Deontic Logic <i>Robert Demolombe, Philippe Bretier and Vincent Louis</i>	751
Adaptive Multi-Agent Programming in GTGolog <i>Alberto Finzi and Thomas Lukasiewicz</i>	753
Formalizing Complex Task Libraries in Golog <i>Alfredo Gabaldon</i>	755
Automated Deduction for Logics of Default Reasoning <i>Laura Giordano, Valentina Gliozzi, Nicola Olivetti and Gian Luca Pozzato</i>	757
Reasoning About Motion Patterns <i>Björn Gottfried</i>	759
Applying OPRMs to Recursive Probability Models <i>Catherine Howard and Markus Stumptner</i>	761
Variable Forgetting in Preference Relations over Propositional Domains <i>Philippe Besnard, Jérôme Lang and Pierre Marquis</i>	763
Two Orthogonal Biases for Choosing the Intensions of Emerging Concepts in Ontology Refinement <i>Francesca A. Lisi and Floriana Esposito</i>	765
Computing Possible and Necessary Winners from Incomplete Partially-Ordered Preferences <i>Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable and Toby Walsh</i>	767
What's a Head Without a Body? <i>Christian Anger, Martin Gebser, Tomi Janhunen and Torsten Schaub</i>	769
Irrelevant Updates of Nonmonotonic Knowledge Bases <i>Ján Šefránek and Jozef Šiška</i>	771
Decision Making in Large-Scale Domains: A Case Study <i>Mikhail Soutchanski, Huy Pham and John Mylopoulos</i>	773
Towards a Logic of Agency and Actions with Duration <i>Nicolas Troquard and Laure Vieu</i>	775
5. Model Based Reasoning	
Better Debugging Through More Abstract Observations <i>Wolfgang Mayer and Markus Stumptner</i>	779
Logic Profiling for Multicriteria Rating on Web Pages <i>Alessandra Mileo</i>	781
An Empirical Analysis of the Complexity of Model-Based Diagnosis <i>Gregory Provan</i>	783

6. Machine Learning

Meta-Clustering Gene Expression Data with Positive Tensor Factorizations <i>Liviu Badea and Doina Tilivea</i>	787
Identifying Inter-Domain Similarities Through Content-Based Analysis of Hierarchical Web-Directories <i>Shlomo Berkovsky, Dan Goldwasser, Tsvi Kuflik and Francesco Ricci</i>	789
Calibrating Probability Density Forecasts with Multi-Objective Search <i>Michael Carney and Pádraig Cunningham</i>	791
Term-Weighting in Information Retrieval Using Genetic Programming: A Three Stage Process <i>Ronan Cummins and Colm O'Riordan</i>	793
Adaptation Knowledge Discovery from a Case Base <i>Mathieu d'Aquin, Fadi Badra, Sandrine Lafrogne, Jean Lieber, Amadeo Napoli and Laszlo Szathmary</i>	795
Polynomial Conditional Random Fields for Signal Processing <i>Trinh-Minh-Tri Do and Thierry Artières</i>	797
Stream Clustering Based on Kernel Density Estimation <i>Stefano Lodi, Gianluca Moro and Claudio Sartori</i>	799
Version Space Learning for Possibilistic Hypotheses <i>Henri Prade and Mathieu Serrurier</i>	801
Ensembles of Grafted Trees <i>Juan J. Rodríguez and Jesús M. Maudes</i>	803
A Compression-Based Method for Stemmatic Analysis <i>Teemu Roos, Tuomas Heikkilä and Petri Myllymäki</i>	805
Patch Learning for Incremental Classifier Design <i>Rudy Sicard, Thierry Artières and Eric Petit</i>	807
Version Space Support Vector Machines <i>Evgueni Smirnov, Ida Sprinkhuizen-Kuyper, Georgi Nalbantov and Stijn Vanderlooy</i>	809
Meta-Typicalness Approach to Reliable Classification <i>Evgueni Smirnov, Stijn Vanderlooy and Ida Sprinkhuizen-Kuyper</i>	811
Text Sampling and Re-Sampling for Imbalanced Authorship Identification Cases <i>Efstathios Stamatatos</i>	813
Is Web Genre Identification Feasible? <i>Benno Stein and Sven Meyer zu Eissen</i>	815

7. Natural Language Processing

Adaptive Context-Based Term (Re)Weighting: An Experiment on Single-Word Question Answering <i>Marco Ernandes, Giovanni Angelini, Marco Gori, Leonardo Rigutini and Franco Scarselli</i>	819
How to Analyze Free Text Descriptions for Recommending TV Programmes? <i>Bernd Ludwig and Stefan Mandl</i>	821
Soft Uncoupling of Markov Chains for Permeable Language Distinction: A New Algorithm <i>Richard Nock, Pascal Vaillant, Frank Nielsen and Claudia Henry</i>	823

Tools for Text Mining over Biomedical Literature <i>Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand and Michael Hess</i>	825
SUMMaR: Combining Linguistics and Statistics for Text Summarization <i>Manfred Stede, Heike Bieler, Stefanie Dipper and Arthit Suriyawongkul</i>	827
Phonetic Spelling and Heuristic Search <i>Benno Stein and Daniel Curatolo</i>	829
8. PAIS	
CBR-TM: A New Case-Based Reasoning System for Help-Desk Environments <i>Juan Ángel García-Pardo, Stella Heras Barberá, Rafael Ramos-Garijo, Alberto Palomares, Vicente Julián, Miguel Rebollo and Vicent Botti</i>	833
Verification of Medical Guidelines Using Task Execution with Background Knowledge <i>Arjen Hommersom, Perry Groot, Peter Lucas, Michael Balser and Jonathan Schmitt</i>	835
9. Planning and Scheduling	
The Incompleteness of Planning with Volatile External Information <i>Tsz-Chiu Au and Dana Nau</i>	839
Cost-Optimal Symbolic Planning with State Trajectory and Preference Constraints <i>Stefan Edelkamp</i>	841
A Cooperative Distributed Problem Solving Technique for Large Markov Decision Processes <i>Abdel-Illah Mouaddib and Simon Le Gloannec</i>	843
Integrating Off-Line and On-Line Schedulers <i>Riccardo Rasconi, Nicola Policella and Amedeo Cesta</i>	845
Environment-Driven Skeletal Plan Execution for the Medical Domain <i>Peter Votruba, Andreas Seyfang, Michael Paesold and Silvia Miksch</i>	847
Time Constrained VRP: An Agent Environment-Perception Model <i>Mahdi Zargayouna</i>	849
10. Robotics/Perception	
On Packing 2D Irregular Shapes <i>Alexandros Bouganis and Murray Shanahan</i>	853
Aliasing Maps for Robot Global Localization <i>Emanuele Frontoni and Primo Zingaretti</i>	855
On Interfacing with an Ubiquitous Robotic System <i>Donatella Guarino and Alessandro Saffiotti</i>	857
Leaf Classification Using Navigation-Based Skeletons <i>Georgios Sakellariou and Murray Shanahan</i>	859
Author Index	861

I. Invited Talks

This page intentionally left blank

Socially Intelligent Robots

Cynthia Breazeal, MIT Media Lab, MA, USA

No longer restricted to the factory floor or hazardous environments, autonomous robots are making their way into human environments. Although current commercial examples of domestic robots are more akin to toys, smart appliances, or supervised tools, the need to help ordinary people as capable partners and interact with them in a socially appropriate manner poses new challenges and opens new opportunities for robot applications in the home, office, school, entertainment locales, healthcare institutions, and more. Many of these applications require robots to play a long-term role in people's daily lives.

Developing robots with social and emotional intelligence is a critical step towards enabling them to be intelligent and capable in their interactions with humans, intuitive to communicate with people, able to work cooperatively with people, and able to learn quickly and effectively from natural human instruction.

This talk presents recent progress in these areas and outlines several “grand challenge” problems of social robotics. Specific research projects and applications are highlighted to illustrate how robots with social capabilities are being developed to learn, assist, entertain, or otherwise benefit their human counterparts.

Managing Diversity in Knowledge

Fausto Giunchiglia, Università di Trento, Italy

We are facing an unforeseen growth of the complexity of (data, content and) knowledge. Here we talk of complexity meaning the size, the sheer numbers, the spatial and temporal pervasiveness of knowledge, and the unpredictable dynamics of knowledge change, unknown at design time but also at run time. The obvious example is the Web and all the material, also multimedia, which is continuously made available on line.

Our goal in this talk is to propose a novel approach which deals with this level of complexity and that, hopefully, will overcome some of the scalability issues shown by the existing data and knowledge representation technology. The key idea is to propose a bottom-up approach where diversity is considered as a feature which must be maintained and exploited and not as a defect that must be absorbed in some general schema. The proposed solution amounts to making a paradigm shift from the view where knowledge is mainly assembled by combining basic building blocks to a view where new knowledge is obtained by the design or run-time adaptation of existing knowledge.

Typically, we will build knowledge on top of a landscape of existing highly interconnected knowledge parts. Knowledge will no longer be produced ab initio, but more and more as adaptations of other, existing knowledge parts, often performed in runtime as a result of a process of evolution. This process will not always be controlled or planned externally but induced by changes perceived in the environment in which systems are embedded. The challenge is to develop design methods and tools that enable effective design by harnessing, controlling and using the effects of emergent knowledge properties. This leads to the proposal of developing adaptive and, when necessary, self-adaptive knowledge systems and to the proposal of developing a new methodology for knowledge engineering and management, that we call "Managing Diversity in Knowledge by Adaptation".

We will present and discuss the ideas above considering, as an example, the use of ontologies in the formalization of knowledge (for instance in the SemanticWeb).

The Truth about Defaults

Hector Levesque, University of Toronto, Canada

Virtually all of the work on defaults in AI has concentrated on default reasoning: given a theory T containing facts and defaults of some sort, we study how an ideal agent should reason with T , typically in terms of constructs like fixpoints, or partial orders, or nonmonotonic entailment relationships.

In this talk, we investigate a different question: under what conditions should we consider the theory T to be true, or believed to be true, or all that is believed to be true? By building on truth in this way, we end up with a logic of defaults that is classical, that is, a logic with an ordinary monotonic notion of entailment. And yet default reasoning emerges naturally from these ideas.

We will show how to characterize the default logic of Reiter and the autoepistemic logic of Moore in purely truth-theoretic terms. We will see that the variant proposed by Konolige is in fact a link between the two, and that all three fit comfortably within a single logical language, that we call O3L. Finally, we will present first steps towards a proof theory (with axioms and rules of inference) for O3L. Among other things, this allows us to present ordinary sentence-by-sentence derivations that correspond to different sorts of default reasoning.

This is joint work with Gerhard Lakemeyer.

SmartWeb: Getting Answers on the Go

Wolfgang Wahlster, Universität des Saarlandes - DFKI, Saarbrücken, Germany

The appeal of being able to ask a spoken question to a mobile internet terminal and receive an audible answer immediately has been renewed by the broad availability of always-on Web access, which allows users to carry the internet in their pockets. Ideally, a multimodal dialogue system that uses the Web as its knowledge base would be able to answer a broad range of spoken questions. Practically, the size and dynamic nature of the Web and the fact that the content of most web pages is encoded in natural language makes this an extremely difficult task.

Recent progress in semantic web technology is enabling innovative open-domain question answering engines that use the entire Web as their knowledge base and offer much higher retrieval precision than current web search engines. Our SmartWeb system is based on the fortunate confluence of three major research efforts that have the potential of forming the basis for the next generation of Web-based answer engines. The first effort is the semantic Web, which provides the formalisms, tools and ontologies for the explicit markup of the content of Web pages; the second effort is the development of semantic Web services, which results in a Web where programs act as autonomous agents to become the producers and consumers of information and enable automation of transactions. The third important effort is information extraction from huge volumes of rich text corpora available on the web exploiting language technology and machine learning.

SmartWeb provides a context-aware user interface, so that it can support the mobile user in different roles, e.g. as a car driver, a motor biker, a pedestrian or a sports spectator. One of the demonstrators of SmartWeb is a personal guide for the 2006 FIFA world cup in Germany, that provides mobile infotainment services to soccer fans, anywhere and anytime.

This talk presents the anatomy of SmartWeb and explains the distinguishing features of its multimodal interface and its answer engine. In addition, the talk gives an outlook on the French-German mega project Quaero and its relation to SmartWeb.

II. Papers

This page intentionally left blank

1. Cognitive Modelling

This page intentionally left blank

Background default knowledge and causality ascriptions

Jean-François Bonnefon and Rui Da Silva Neves¹ and Didier Dubois and Henri Prade²

Abstract. A model is defined that predicts an agent's ascriptions of causality (and related notions of facilitation and justification) between two events in a chain, based on background knowledge about the normal course of the world. Background knowledge is represented by nonmonotonic consequence relations. This enables the model to handle situations of poor information, where background knowledge is not accurate enough to be represented in, e.g., structural equations. Tentative properties of causality ascriptions are explored, i.e., preference for abnormal factors, transitivity, coherence with logical entailment, and stability with respect to disjunction and conjunction. Empirical data are reported to support the psychological plausibility of our basic definitions.

1 INTRODUCTION

Models of causal ascriptions crucially depend on the choice of an underlying representation for the causality-ascribing agent's knowledge. Unlike standard diagnosis problems (wherein an unobserved cause must be inferred from observed events and known causal links), causality ascription is a problem of describing as 'causal' the link between two observed events in a sequence. The first step in modeling causal ascription is to define causality in the language chosen for the underlying representation of knowledge. In this article, we define and discuss a model of causal ascription that represents knowledge by means of nonmonotonic consequence relations.³ Indeed, agents often must cope with poor knowledge about the world, under the form of default rules. Clearly, this type of background knowledge is less accurate than, e.g., structural equations. It is nevertheless appropriate to predict causal ascriptions in situations of restricted knowledge. Section 2 presents the logical language we will use to represent background knowledge. Section 3 defines our main notions of causality and facilitation ascriptions. Empirical data are reported to support the distinction between these two notions. Section 4 establishes some formal properties of the model. Section 5 distinguishes the notion of epistemic justification from that of causality. Section 6 relates our model to other works on causality in AI.

2 MODELING BACKGROUND KNOWLEDGE

The agent is supposed to have observed or learned of a sequence of events, e.g.: $\neg B_t, A_t, B_{t+1}$. This expresses that B was false at time t , when A took place, and that B became true afterwards ($t + 1$ denotes a time point after t). There is no uncertainty about these events.

Besides, the agent maintains a knowledge-base made of conditional statements of the form 'in context C , if A takes place then B '

¹ LTC-CNRS and DSVP, respectively, 5 allées A. Machado 31058 Toulouse Cedex 9, France. E-mail: {bonnefon,neves}@univ-tlse2.fr.

² IRIT-CNRS, 118 Route de Narbonne, 31062 Toulouse Cedex, France. E-mail: {dubois,prade}@irit.fr

³ This model was advocated in a recent workshop paper [9].

is generally true afterwards', or 'in context C , B is generally true'. These will be denoted by $A_t \wedge C_t \succsim B_{t+1}$, and by $C_t \succsim B_t$, respectively. (Time indices will be omitted when there is no risk of confusion.) The conditional beliefs of an agent with respect to B when an action A takes place or not in context C can take three forms: (i) If A takes place B is generally true afterwards: $A_t \wedge C_t \succsim B_{t+1}$; (ii) If A takes place B is generally false afterwards: $A_t \wedge C_t \succsim \neg B_{t+1}$; (iii) If A takes place, one cannot say whether B is generally true or false afterwards: $A_t \wedge C_t \not\succsim B_{t+1}$ and $A_t \wedge C_t \not\succsim \neg B_{t+1}$.

We assume that the nonmonotonic consequence relation \succsim satisfies the requirements of 'System P' [18]; namely, \succsim is reflexive and the following postulates and characteristic properties hold (\models denotes classical logical entailment):

<i>Left Equivalence</i>	$E \succsim G$ and $E \equiv F$	imply	$F \succsim G$
<i>Right Weakening</i>	$E \succsim F$ and $F \models G$	imply	$E \succsim G$
<i>AND</i>	$E \succsim F$ and $E \succsim G$	imply	$E \succsim F \wedge G$
<i>OR</i>	$E \succsim G$ and $F \succsim G$	imply	$E \vee F \succsim G$
<i>Cautious Monotony</i>	$E \succsim F$ and $E \succsim G$	imply	$E \wedge F \succsim G$
<i>Cut</i>	$E \succsim F$ and $E \wedge F \succsim G$	imply	$E \succsim G$

In addition, we assume $\not\succsim$ to obey the property of Rational Monotony, a strong version of Cautious Monotony[19]:

$$\text{Rational Monotony} \quad E \not\succsim \neg F \text{ and } E \succsim G \quad \text{imply} \quad E \wedge F \succsim G$$

Empirical studies repeatedly demonstrated [1, 2, 5, 10, 24] that System P and Rational Monotony provide a psychologically plausible representation of background knowledge and default inference. Arguments for using nonmonotonic logics in modeling causal reasoning were also discussed in the cognitive science literature [26].

3 ASCRIBING CAUSALITY OR FACILITATION

In the following definitions, A , B , C , and F are either reported actions or statements describing states of affairs, even though notations do not discriminate between them, since the distinction does not yet play a crucial role in the model. When nothing takes place, the persistence of the truth status of statements is assumed in the normal course of things, i.e., $B_t \succsim B_{t+1}$ and $\neg B_t \succsim \neg B_{t+1}$.

Assume that in a given context C , the occurrence of event B is known to be exceptional (i.e., $C \succsim \neg B$). Assume now that F and A are such that $F \wedge A \not\succsim \neg B$ on the one hand, and $A \wedge F \wedge C \succsim B$ on the other hand; we will say that in context C , A together with F are perceived as the *cause* of B (denoted $C : A \wedge F \Rightarrow_{ca} B$), while F alone is merely perceived to have *facilitated* the occurrence of B (denoted $C : F \Rightarrow_{fa} B$).

Definition 1 (Facilitation ascription). An agent that, in context C , learns of the sequence $\neg B_t, F_t, B_{t+1}$ will judge that $C : F \Rightarrow_{fa} B$ if it believes that $C \succsim \neg B$, and that both $F \wedge A \not\succsim \neg B$ and $F \wedge C \not\succsim \neg B$.

Definition 2 (Causality ascription). An agent that, in context C , learns of the sequence $\neg B_t, A_t, B_{t+1}$ will judge that $C : A \Rightarrow_{ca} B$ if it believes that $C \vdash \neg B$, and $A \wedge C \vdash B$.

Example 1 (Driving while intoxicated). When driving, one has generally no accident, $Drive \vdash \neg Accident$. This is no longer true when driving while drunk, which is not as safe, $Drive \wedge Drunk \not\vdash \neg Accident$; moreover, fast driving while drunk will normally lead to an accident, $Drive \wedge Fast \wedge Drunk \vdash Accident$. Suppose now that an accident took place after the driver drove fast while being drunk. $Fast \wedge Drunk$ will be perceived as the cause of the accident, while $Drunk$ will only be judged as having facilitated the accident.

Note that Def. 1 is weaker than saying F ‘prevents’ $\neg B$ from persisting: $\not\vdash$ does not allow the jump from ‘not having $\neg B$ ’ to ‘ B ’. In Def. 2, the fact that B is exceptional in context C precludes the possibility for C to be the cause of B – but not the possibility that $B \models C$, i.e., that C is a necessary condition of B . Thus, context can be a necessary condition of B without being perceived as its cause.

An interesting situation arises when an agent only knows that $C \vdash \neg B$ and $F \wedge C \not\vdash \neg B$, and learns of the sequence of events $\neg B_t$ (in context C), F_t , B_{t+1} . Although this situation should lead the agent to judge that $C : F_t \Rightarrow_{fa} B_{t+1}$, it may be tempting to judge that $C : F_t \Rightarrow_{ca} B_{t+1}$, as long as no other potential cause reportedly took place. Another interesting situation arises when, in context C , an agent learns of the sequence $\neg B_t, A_t$, and B_{t+1} , while it believes that $\neg B_t \wedge C \vdash \neg B_{t+1}$, and that $A_t \wedge C \vdash \neg B_{t+1}$. Then the agent cannot consider that $C : A_t \Rightarrow_{ca} B_{t+1}$, and it may suspect some fact went unreported: finding about it would amount to a diagnosis problem.

There is no previous empirical support to the distinction we introduce between ascriptions of cause and facilitation. To check whether this distinction has intuitive appeal to lay reasoners, we conducted two experiments in which we presented participants with different sequences of events. We assessed their relevant background knowledge, from which we predicted the relations of cause and facilitation they should ascribe between the events in the sequence. We then compared these predictions to their actual ascriptions.

3.1 Experiment 1

Methods Participants were 46 undergraduate students. None was trained in formal logic or in philosophy. Participants read the stories of three characters, and answered six questions after reading each story. The three characters were described as constantly feeling very tired (an uncommon feeling for them) after two recent changes in their lives: working at night and having a stressful boss (for the first character), working at night and becoming a dad (for the second character), and having a stressful boss and becoming a dad (for the third character). The first three questions assessed participants’ background knowledge with respect to (i) the relation between the first event and feeling constantly tired; (ii) the second event and feeling constantly tired; and (iii) the conjunction of the two events and feeling constantly tired. For example:

What do you think is the most common, the most normal: Working at night and feeling constantly tired, or working at night and not feeling constantly tired? or are those equally common and normal?

Participants who chose the first, second, and third answer were assumed to endorse $WorkNight \vdash Tired$; $WorkNight \vdash \neg Tired$; and $(WorkNight \not\vdash Tired) \wedge (WorkNight \not\vdash \neg Tired)$, respectively. The fourth, fifth, and sixth questions assessed participants’ ascriptions of causality or facilitation between (i) the first event and feeling constantly tired; (ii) the second event and feeling constantly tired; and

(iii) the conjunction of the two events and feeling constantly tired. E.g., one of these questions read:

Fill in the blank with the word ‘caused’ or ‘facilitated’, as seems the most appropriate. If neither seems appropriate, fill in the blank with ‘xxx’: Working at night ... the fact that Julien feels constantly tired.

Results Out of the 116 ascriptions that the model predicted to be of facilitation, 68% indeed were, 11% were of causality, and 21% were neither. Out of the 224 ascriptions that the model predicted to be of causality, 46% indeed were, 52% were of facilitation, and 2% were neither. The global trend in the results is thus that background knowledge that theoretically matches a facilitation ascription indeed largely leads people to make such an ascription, while background knowledge that theoretically matches a causality ascription leads people to divide equally between causality and facilitation ascriptions. This trend is statistically reliable for almost all ascriptions required by the task. Relevant statistics (χ^2 scores) are higher than 7.7 for 7 out of the 9 ascriptions ($p < .05$, one-tailed, in all cases), and higher than 3.2 for the remaining two ascriptions ($p < .10$, one-tailed, in both cases). From these results, it appears that the notion of facilitation does have intuitive appeal to lay reasoners, and that it is broadly used as defined in our model. In particular, it clearly has a role to play in situations where an ascription of causality sounds too strong a conclusion, but no ascription at all sounds too weak.

3.2 Experiment 2

Experiment 2 was designed to consolidate the results of Experiment 1 and to answer the following questions: Does the fact that background knowledge match Def. 1 or Def. 2 affect the strength of the link participants perceive between two reported events, and does this perceived strength in turn determine whether they make an ascription of causality or facilitation?

Methods Participants were 41 undergraduates. Elements of their background knowledge were assessed as in Exp. 1, in order to select triples of propositions $<Context, Factor, Effect>$ that matched either Def. 1 or Def. 2. E.g., a participant might believe that one has generally no accident when driving, but that one will generally have an accident when driving after some serious drinking; for this participant, $<Drive, SeriousDrinking, Accident>$ is a match with Def. 2. Participants then rated on a 9-point scale how strongly *Factor* and *Effect* were related. Finally, as a measure of ascription, they chose an appropriate term to describe the relation between *Factor* and *Effect*, from a list including ‘causes’ and ‘facilitates’.

Results Out of the 16 ascriptions that the model predicted to be of facilitation, 14 were so, and 2 were of causality. Out of the 25 ascriptions that the model predicted to be of causality, 11 were so, and 14 were of facilitation. Beliefs thus had the expected influence on ascriptions, $\chi^2 = 4.5$, $p < .05$. The trend observed in Experiment 1 is replicated in Experiment 2. We also conducted a *mediation analysis* of our data, which consists in a series of 3 regression analyzes. The direct effect of background knowledge on ascription was significant, $\beta = .33$, $p < .05$. The effect of background knowledge on perceived strength was also significant, $\beta = .41$, $p < .01$. In the third regression, background knowledge and perceived strength were entered simultaneously. Perceived strength was a reliable predictor of ascription, $\beta = .29$, $p < .05$, which was no longer the case for background knowledge, $\beta = .23$, $p > .05$. Data thus meet the requirement

of a mediational effect: Whether the background knowledge of participants matches Def. 1 or Def. 2 determines their final ascription of $C : \text{Factor} \Rightarrow_{\text{fa}} \text{Effect}$ or $C : \text{Factor} \Rightarrow_{\text{ca}} \text{Effect}$ through its effect on the perceived strength of the link between *Factor* and *Effect*.

4 PROPERTIES OF CAUSAL ASSCRIPTIONS

4.1 Impossibility of mutual causality

Proposition 1. *If $C : A \Rightarrow_{\text{ca}} B$, then it cannot hold that $C : B \Rightarrow_{\text{ca}} A$.*

Proof. If $C : A \Rightarrow_{\text{ca}} B$, it holds that $C \succ \neg A$, $C \wedge A \succ B$, and the sequence $\neg B_i, A_i, B_{i+1}$ has been observed. This is not inconsistent with $C \succ \neg A$, $C \wedge B \succ A$ (the background knowledge part of $C : B \Rightarrow_{\text{ca}} A$), but it is inconsistent with the sequence $\neg A_i, B_i, A_{i+1}$ that would allow the ascription $C : B \Rightarrow_{\text{ca}} A$. \square

4.2 Preference for abnormal causes

Psychologists established that abnormal conditions are more likely to be selected by human agents as the cause of an event [17] and more so if this event is itself abnormal [12] (see also [16] in the area of legal philosophy). Our model reflects this preference: Only what is abnormal in a given context can be perceived as facilitating or causing a change in the normal course of things in this context.

Proposition 2. *If $C : A \Rightarrow_{\text{ca}} B$ or $C : A \Rightarrow_{\text{fa}} B$, then $C \succ \neg A$.*

Proof. $C \succ \neg A$ is false when either $C \succ A$ or $C \not\succ \neg A$. If $C \succ A$, it cannot be true that both $C \succ \neg B$ and either $A \wedge C \not\succ \neg B$ (the definition of $C : A \Rightarrow_{\text{fa}} B$) or $A \wedge C \succ B$ (the definition of $C : A \Rightarrow_{\text{ca}} B$). This is due to the Cautious Monotony property of \succ , which forces $C \wedge A \succ \neg B$ from $C \succ A$ and $C \succ \neg B$. Likewise, the Rational Monotony of \succ forces $C \wedge A \succ \neg B$ from $C \not\succ \neg A$ and $C \succ \neg B$; thus, it cannot be the case that $C : A \Rightarrow_{\text{fa}} B$ or $C : A \Rightarrow_{\text{ca}} B$ when $C \not\succ \neg A$. \square

Example 2 (The unreasonable driver). *Let us imagine an agent who believes it is normal to be drunk in the context of driving ($\text{Drive} \succ \text{Drunk}$). This agent may think that it is exceptional to have an accident when driving ($\text{Drive} \succ \neg \text{Accident}$). In that case, the agent cannot but believe that accidents are exceptional as well when driving while drunk: $\text{Drive} \wedge \text{Drunk} \succ \neg \text{Accident}$. As a consequence, when learning that someone got drunk, drove his car, and had an accident, this agent will neither consider that $C : \text{Drunk} \Rightarrow_{\text{fa}} \text{Accident}$ nor that $C : \text{Drunk} \Rightarrow_{\text{ca}} \text{Accident}$.*

4.3 Transitivity

Def. 2 does not grant general transitivity to \Rightarrow_{ca} . If $C : A \Rightarrow_{\text{ca}} B$ and $C : B \Rightarrow_{\text{ca}} D$, it does not always follow that $C : A \Rightarrow_{\text{ca}} D$. Formally: $C \succ \neg B$ and $A \wedge C \succ B$ and $C \succ \neg D$ and $B \wedge C \succ D$ do not entail $C \succ \neg D$ and $A \wedge C \succ D$, because \succ itself is not transitive. Although \Rightarrow_{ca} is not generally transitive, it becomes so in one particular case.

Proposition 3. *If $C : A \Rightarrow_{\text{ca}} B$, $C : B \Rightarrow_{\text{ca}} D$, and $B \wedge C \succ A$, then $C : A \Rightarrow_{\text{ca}} D$.*

Proof. From the definition of $C : B \Rightarrow_{\text{ca}} D$, it holds that $B \wedge C \succ D$. From $B \wedge C \succ A$ and $B \wedge C \succ D$, applying Cautious Monotony yields $A \wedge B \wedge C \succ D$, which together with $A \wedge C \succ B$ (from the definition of $C : A \Rightarrow_{\text{ca}} B$) yields by Cut $A \wedge C \succ D$; since it holds from the definition of $C : B \Rightarrow_{\text{ca}} D$ that $C \succ \neg D$, the two parts of the definition of $C : A \Rightarrow_{\text{ca}} D$ are satisfied. \square

Example 3 (Mud on the plates). *Driving back from the countryside, you get a fine because your plates are muddy, $\text{Drive} : \text{Mud} \Rightarrow_{\text{ca}} \text{Fine}$. Let us assume that you perceive your driving to the countryside as the cause for the plates to be muddy, $\text{Drive} : \text{CountrySide} \Rightarrow_{\text{ca}} \text{Mud}$. For transitivity to apply, i.e., to judge that $\text{Drive} : \text{CountrySide} \Rightarrow_{\text{ca}} \text{Fine}$, it must hold that $\text{Mud} \wedge \text{Drive} \succ \text{CountrySide}$: If mud on your plates usually means that you went to the countryside, then the trip can be considered the cause of the fine. If the presence of mud on your plates does not allow to infer that you went to the countryside (perhaps you also regularly drive through muddy streets where you live), then transitivity is not applicable; you will only consider that the mud caused the fine, not that the trip did.*

4.4 Entailment and causality ascriptions

Classical entailment \models does not preserve \Rightarrow_{ca} . If $C : A \Rightarrow_{\text{ca}} B$ and $B \models B'$, one cannot say that $C : A \Rightarrow_{\text{ca}} B'$. Indeed, while $A \wedge C \succ B$ follows by right weakening [18] from $A \wedge C \succ B$, it is not generally true that $C \succ \neg B'$, given that $C \succ \neg B$. Besides, according to Definition 2, if $A' \models A$, the fact that $C : A \Rightarrow_{\text{ca}} B$ does not entail that $C : A' \Rightarrow_{\text{ca}} B$, since $C \succ \neg B$ and $A \wedge C \succ B$ do not entail $A' \wedge C \succ B$ when $A' \models A$. This fact is due to the extreme cautiousness of System P. It is contrasted in the following example with Rational Monotony.

Example 4 (Stone throwing). *An agent believes that a window shattered because a stone was thrown at it ($\text{Window} : \text{Stone} \Rightarrow_{\text{ca}} \text{Shatter}$), based on its beliefs that $\text{Window} \succ \neg \text{Shatter}$ and $\text{Stone} \wedge \text{Window} \succ \text{Shatter}$. Using the Cautious Monotony of System P, it is not possible to predict that the agent would make a similar ascription if a small stone had been thrown (SmallStone), or even if a white stone had been thrown (WhiteStone), or even if a big stone had been thrown (BigStone), although it holds that $\text{SmallStone} \models \text{Stone}$, $\text{WhiteStone} \models \text{Stone}$, and $\text{BigStone} \models \text{Stone}$. Adding Rational Monotony [19] to System P allows the ascriptions $\text{Window} : \text{BigStone} \Rightarrow_{\text{ca}} \text{Shatter}$ and $\text{Window} : \text{WhiteStone} \Rightarrow_{\text{ca}} \text{Shatter}$, but also $\text{Window} : \text{SmallStone} \Rightarrow_{\text{ca}} \text{Shatter}$. To block this last ascription, it would be necessary that the agent has specific knowledge about the harmlessness of small stones, such as $\text{Window} \wedge \text{Smallstone} \not\models \text{Shatter}$ or even $\text{Window} \wedge \text{Smallstone} \succ \neg \text{Shatter}$.*

4.5 Stability w.r.t. disjunction and conjunction

\Rightarrow_{ca} is stable with respect to disjunction, both on the right and on the left, and stable w.r.t. conjunction on the right.

Proposition 4. *The following properties hold:*

1. *If $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$, then $C : A \Rightarrow_{\text{ca}} B \vee B'$.*
2. *If $C : A \Rightarrow_{\text{ca}} B$ and $C : A' \Rightarrow_{\text{ca}} B$, then $C : A \vee A' \Rightarrow_{\text{ca}} B$.*
3. *If $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$, then $C : A \Rightarrow_{\text{ca}} B \wedge B'$.*

Proof. Applying AND to the first part of the definitions of $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$, i.e., $C \succ \neg B$ and $C \succ \neg B'$, yields $C \succ \neg B \wedge \neg B'$, and thus $C \succ \neg(B \vee B')$. Now, applying AND to the second part of the definitions of $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$, i.e., $A \wedge C \succ B$ and $A \wedge C \succ B'$, yields $A \wedge C \succ B \wedge B'$, which together with Right Weakening yields $A \wedge C \succ B \vee B'$. The definition of $C : A \Rightarrow_{\text{ca}} B \vee B'$ is thus satisfied. The proof of Fact 2 is obtained by applying OR to the second part of the definitions of $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$. Finally, applying AND to the first part of the definitions of $C : A \Rightarrow_{\text{ca}} B$ and $C : A \Rightarrow_{\text{ca}} B'$, i.e., $C \succ \neg B$ and $C \succ \neg B'$, yields $C \succ \neg B \wedge \neg B'$, which together with Right Weakening, yields

$C \vdash \neg B \vee \neg B'$, and thus $C \vdash \neg(B \wedge B')$. Now, applying AND to the second part of the definitions of $C : A \Rightarrow_{ca} B$ and $C : A \Rightarrow_{ca} B'$, i.e., $A \wedge C \vdash B$ and $A \wedge C \vdash B'$, yields $A \wedge C \vdash B \wedge B'$. The definition of $C : A \Rightarrow_{ca} B \wedge B'$ is thus satisfied. \square

\Rightarrow_{ca} is not stable w.r.t. conjunction on the left. If $C : A \Rightarrow_{ca} B$ and $C : A' \Rightarrow_{ca} B$, then it is not always the case that $C : A \wedge A' \Rightarrow_{ca} B$ (see example 5). This lack of stability is once again due to the cautiousness of System P; for $C : A \wedge A' \Rightarrow_{ca} B$ to hold, it is necessary that $C \wedge A \vdash A'$ or, alternatively, that $C \wedge A' \vdash A$. Then Cautious Monotony will yield $A \wedge A' \wedge C \vdash B$. Rational Monotony can soften this constraint and make it enough that $C \wedge A \not\vdash \neg A'$ or $C \wedge A' \not\vdash \neg A$.

Example 5 (Busy professors). Suppose that professors in your department seldom show up early at the office ($Prof \vdash \neg Early$). However, they generally do so when they have tons of student papers to mark ($Prof \wedge Mark \vdash Early$), and also when they have a grant proposal to write ($Prof \wedge Grant \vdash Early$). When learning that a professor had tons of papers to grade and that she came in early, you would judge that $Prof : Mark \Rightarrow_{ca} Early$. Likewise, when learning that a professor had a grant proposal to write and came in early, you would judge that $Prof : Grant \Rightarrow_{ca} Early$. But what if you learn that a professor had tons of papers to grade and a grant proposal to write and that she came in early? That would depend on whether it is an exceptional situation to have to deal with both tasks on the same day. If it is not exceptional ($Mark \not\vdash \neg Grant$), then you will judge that $Prof : Mark \wedge Grant \Rightarrow_{ca} Early$. If, on the contrary, $Mark \wedge Grant$ is an exceptional event, it does not hold anymore that $Mark \wedge Grant \vdash Early$, and it is thus impossible to feel sure about $Prof : Mark \wedge Grant \Rightarrow_{ca} Early$. For example, it might be the case that faced with such an exceptional workload, a professor will prefer working at home all day rather than coming to the office. In that case, her coming in early would be due to another factor, e.g., a meeting that could not be cancelled.

5 ASRIPTIONS OF JUSTIFICATION

Perceived causality as expressed in Def. 2 should be distinguished from the situation that we term ‘justification.’ We write that $C : A \Rightarrow_{ju} B$ when an agent judges that the occurrence of A in context C gave reason to expect the occurrence of B .

Definition 3 (Justification). An agent that learns in context C of the sequence $\neg B_t, A_t, B_{t+1}$ will judge that $C : A \Rightarrow_{ju} B$ if it believes that $C \not\vdash \neg B, C \not\vdash B$ and $A \wedge C \vdash B$.

Faced with facts $C, \neg B_t, A_t, B_{t+1}$, an agent believing that $C \not\vdash \neg B, C \not\vdash B$ and $A \wedge C \vdash B$ may doubt that the change from $\neg B_t$ to B_{t+1} is really due to A_t , although the latter is indeed the very reason for the lack of surprise at having B_{t+1} reported. Indeed, situation $\neg B_t$ at time t appears to the agent to be contingent, since it is neither a normal nor an abnormal course of things in context C . This clearly departs from the situation where $C \vdash \neg B$ and $A \wedge C \vdash B$, wherein the agent will judge that $C : A \Rightarrow_{ca} B$. In a nutshell, the case whereby $C \not\vdash \neg B, C \not\vdash B$ and $A \wedge C \vdash B$ cannot be interpreted as the recognition of a causal phenomenon by an agent: All that can be said is that reporting A caused the agent to start believing B , and that she should not be surprised of having B_{t+1} reported.

What we call justification is akin to the notion of explanation following Spohn [27]: Namely, ‘ A is a reason for B ’ when raising the epistemic rank for A raises the epistemic rank for B . Gärdenfors [11] captured this view to some extent, assuming that A is a reason for B if B is not retained in the contraction of A . Williams et al.

[29] could account for the Spohnian view in a more refined way using kappa-rankings and transmutations, distinguishing between weak and strong explanations. As our framework can easily be given a possibilistic semantics [4], it could properly account for this line of thought, although our distinction between perceived causation and epistemic justification is not the topic of the above works.

6 RELATED WORKS

Causality plays a central role in at least two problems studied in AI, diagnosis and the simulation of dynamical systems. Diagnosis problems are a matter of abduction: One takes advantage of the knowledge of some causal links to infer the most plausible causes of an observed event [23]. In this setting, causality relations are often modelled by conditional probabilities $P(\text{effect}|\text{cause})$.⁴ Dynamical systems are modelled in AI with respect, e.g., to qualitative physics [6], and in logics of action. The relation of nonmonotonic inference to causality has already been emphasized by authors dealing with reasoning about actions and the frame problem [13, 20, 28]. Material implication being inappropriate to represent a causal link, these approaches define a ‘causal rule’ as ‘there is a cause for effect B to be true if it is true that A has just been executed’, where ‘there is a cause for’ is modelled by a modal operator.

The problem discussed in this paper is not, however, one of classical diagnosis. Neither does it deal with the qualitative simulation of dynamical systems, nor with the problem of describing changes caused by the execution of actions, nor with what does not change when actions are performed. We are concerned here with a different question, namely the explanation of a sequence of reported events, in terms of pairs of events that can be considered as related by a causality relation. In that sense, our work is reminiscent of the ‘causal logic’ of Shafer [25], which provides a logical setting that aims at describing the possible relations of concomitance between events when an action takes place. However, Shafer’s logic does not leave room for abnormality. This notion is central in our approach, as it directly relates to the relations of qualitative independence explored in [7] – causality and independence being somewhat antagonistic notions.

Following [22], Halpern and Pearl [14, 15] have proposed a model that distinguishes real causes (‘cause in fact’) from potential causes, by using an a priori distinction between ‘endogenous’ variables (the possible values of which are governed by structural equations, for example physical laws), and ‘exogenous’ variables (determined by external factors). Exogenous variables cannot be deemed causal. Halpern and Pearl’s definition of causality formalizes the notion of an active causal process. More precisely, the fact A that a subset of endogenous variables has taken some definite values is the real cause of an event B if (i) A and B are true in the real world, (ii) this subset is minimal, (iii) another value assignment to this subset would make B false, the values of the other endogenous variables that do not directly participate to the occurrence of B being fixed in some manner, and (iv) A alone is enough for B to occur in this context. This approach, thanks to the richness of background knowledge when it is represented in structural equations, makes it possible to treat especially difficult examples. Our model is not to be construed as an alternative or a competitor to models based on structural equations. Indeed, we see our approach as either a ‘plan B’ or a complement to structural equation modeling. One might not have access to the accurate

⁴ Nevertheless, Bayesian networks [21] (that represent a joint probability distribution by means of a directed graph) do not necessarily reflect causal links between their nodes, for different graphical representations can be obtained depending on the ordering in which variables are considered [8].

information needed to build a structural equation model; in this case, our less demanding model might still be operable. Alternatively, a decision support system may be able to build a structural equation model of the situation, although its users only have access to qualitative knowledge. In that case, the system will be able to compare its own causality ascriptions to the conclusions of the qualitative model, and take appropriate explanatory steps, would those ascriptions be too different. Indeed, our model does not aim at identifying the true, objective cause of an event, but rather at predicting what causal ascription an agent would make based on the limited information it has at its disposal.

Models based on structural equations are often supplemented with the useful notion of *intervention*. In many situations, finding the cause of an event will be much easier if the agent can directly intervene in the manner of an experimenter. In future work, we intend to explore the possibility of supplementing our own model with a similar notion by means of a $\text{do}(\bullet)$ operator. An ascription of causality (resp., facilitation) would be made iff the requirements of Definition 2 (resp., 1) are met both for A, B, C and for $\text{do}(A), B, C$, where $\text{do}(A)$ means that the occurrence of A is forced by an intervention [22]. As for now, we only give a brief example of how such an operator can be used in our approach.

Example 6 (Yellow teeth). *An agent learns that someone took up smoking, that this person's teeth yellowed, and that this person developed lung cancer. The agent believes that generally speaking, it is abnormal to be a smoker, to have yellow teeth, and to develop lung cancer (resp., $(C \vdash \neg\text{Smoke}, C \vdash \neg\text{Yellow}, C \vdash \neg\text{Lung})$). The agent believes that it is normal for smokers to have yellow teeth ($C \wedge \text{Smoke} \vdash \text{Yellow}$) and to develop lung cancer ($C \wedge \text{Smoke} \vdash \text{Lung}$), and that it is not abnormal for someone who has yellow teeth to develop lung cancer ($C \wedge \text{Yellow} \not\vdash \neg\text{Lung}$). From these beliefs and observations, Definitions 1 and 2 would allow for various ascriptions, including the following one: Smoking caused the yellow teeth which in turn facilitated lung cancer. With the additional constraint based on the $\text{do}(\bullet)$ operator, only one set of ascriptions remains possible: Both the yellow teeth and the lung cancer were caused by smoking. Yellow teeth cannot be said anymore to facilitate lung cancer because, inasmuch as lung cancer is generally abnormal, it holds that $C \wedge \text{do}(\text{Yellow}) \vdash \neg\text{Lung}$: There is no reason to think that one will develop lung cancer after painting one's teeth yellow.*

7 CONCLUDING REMARKS

We have presented a simple qualitative model of the causal ascriptions an agent will make from its background default knowledge, when confronted with a series of events. In addition to supplementing this model with a $\text{do}(\bullet)$ operator, we intend to extend our present work in three main directions. First, we should be able to equip our framework with possibilistic qualitative counterparts to Bayesian networks [3], since System P augmented with Rational Monotony can be represented in possibilistic logic [4]. Second, we will derive postulates for causality from the independence postulates presented in [7]. Finally, in parallel to further theoretical elaboration, we will maintain a systematic experimental program that will test the psychological plausibility of our definitions, properties, and postulates.

ACKNOWLEDGMENTS

This work was supported by a grant from the Agence Nationale pour la Recherche, project number NT05-3-44479.

REFERENCES

- [1] S. Benferhat, J. F. Bonnefon, and R. M. Da Silva Neves, 'An experimental analysis of possibilistic default reasoning', in *KR2004*, pp. 130–140. AAAI Press, (2004).
- [2] S. Benferhat, J. F. Bonnefon, and R. M. Da Silva Neves, 'An overview of possibilistic handling of default reasoning: An experimental study', *Synthese*, **146**, 53–70, (2005).
- [3] S. Benferhat, D. Dubois, L. Garcia, and H. Prade, 'On the transformation between possibilistic logic bases and possibilistic causal networks', *International Journal of Approximate Reasoning*, **29**, 135–173, (2002).
- [4] S. Benferhat, D. Dubois, and H. Prade, 'Nonmonotonic reasoning, conditional objects and possibility theory', *Artificial Intelligence*, **92**, 259–276, (1997).
- [5] R. M. Da Silva Neves, J. F. Bonnefon, and E. Raufaste, 'An empirical test for patterns of nonmonotonic inference', *Annals of Mathematics and Artificial Intelligence*, **34**, 107–130, (2002).
- [6] J. de Kleer and J. S. Brown, 'Theories of causal ordering', *Artificial Intelligence*, **29**, 33–61, (1986).
- [7] D. Dubois, L. Fariñas Del Cerro, A. Herzig, and H. Prade, *A roadmap of qualitative independence*, volume 15 of *Applied Logic series*, 325–350. Kluwer, Dordrecht, The Netherlands, 1999.
- [8] D. Dubois and H. Prade, 'Probability theory in artificial intelligence. Book review of J. Pearl's "Probabilistic Reasoning in Intelligent Systems"', *Journal of Mathematical Psychology*, **34**, 472–482, (1999).
- [9] D. Dubois and H. Prade, 'Modeling the role of (ab)normality in the ascription of causality judgements by agents', in *NRAC'05*, pp. 22–27, (2005).
- [10] M. Ford, 'System LS: A three tiered nonmonotonic reasoning system', *Computational Intelligence*, **20**, 89–108, (2004).
- [11] P. Gärdenfors, 'The dynamics of belief systems: Foundations vs. coherence theories', *Revue Internationale de Philosophie*, **44**, 24–46, (1990).
- [12] I. Gavansky and G. L. Wells, 'Counterfactual processing of normal and exceptional events', *Journal of Experimental Social Psychology*, **25**, 314–325, (1989).
- [13] E. Giunchiglia, J. Lee, N. McCain, V. Lifschitz, and H. Turner, 'Non-monotonic causal theories', *Artificial Intelligence*, **153**, 49–104, (2004).
- [14] J. Halpern and J. Pearl, 'Causes and explanations: A structural-model approach — part 1: Causes', *British Journal for the Philosophy of Science*, (to appear).
- [15] J. Halpern and J. Pearl, 'Causes and explanations: A structural-model approach — part 2: Explanations', *British Journal for the Philosophy of Science*, (to appear).
- [16] H. L. A. Hart and T. Honoré, *Causation in the law*, Oxford University Press, Oxford, 1985.
- [17] D. J. Hilton and B. R. Slugoski, 'Knowledge-based causal attribution: The abnormal conditions focus model', *Psychological Review*, **93**, 75–88, (1986).
- [18] S. Kraus, D. Lehmann, and M. Magidor, 'Nonmonotonic reasoning, preferential models and cumulative logics', *Artificial Intelligence*, **44**, 167–207, (1990).
- [19] D. Lehmann and M. Magidor, 'What does a conditional knowledge base entails?', *Artificial Intelligence*, **55**, 1–60, (1992).
- [20] N. McCain and H. Turner, 'A causal theory of ramifications and qualifications', in *IJCAI'95*, San Francisco, CA, (1995). Morgan Kaufmann.
- [21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Mateo, CA, 1988.
- [22] J. Pearl, *Causality: Models, Reasoning, and Inference*, Cambridge University Press, Cambridge, 2000.
- [23] Y. Peng and J. A. Reggia, *Abductive Inference Models for Diagnostic Problem-Solving*, Springer Verlag, Berlin, 1990.
- [24] N. Pfeifer and G. D. Kleiter, 'Coherence and nonmonotonicity in human reasoning', *Synthese*, **146**, 93–109, (2005).
- [25] G. Shafer, 'Causal logic', in *ECAI'98*, pp. 711–719, Chichester, England, (1998). Wiley.
- [26] Y. Shoham, 'Nonmonotonic reasoning and causation', *Cognitive Science*, **14**, 213–252, (1990).
- [27] W. Spohn, 'Deterministic and probabilistic reasons and causes', *Erkenntnis*, **19**, 371–393, (1983).
- [28] H. Turner, 'A logic of universal causation', *Artificial Intelligence*, **113**, 87–123, (1999).
- [29] M.-A. Williams, M. Pagnucco, N. Foo, and B. Sims, 'Determining explanations using transmutations', in *IJCAI'95*, pp. 822–830. Morgan Kaufmann, (1995).

Comparing sets of positive and negative arguments: Empirical assessment of seven qualitative rules

Jean-François Bonnefon¹ and Hélène Fargier²

Abstract. Many decisions can be represented as bipolar, qualitative sets of arguments: Arguments can be pros or cons, and ranked according to their importance, but not numerically evaluated. The problem is then to compare these qualitative, bipolar sets. In this paper (a collaboration between a computer scientist and a psychologist), seven procedures for such a comparison are empirically evaluated, by matching their predictions to choices made by 62 human participants on a selection of 33 situations. Results favor cardinality-based procedures, and in particular one that allows for the internal cancellation of positive and negative arguments within a decision.

1 Introduction

Would you rather go to Rome or Beijin next spring? There are many arguments for or against each choice, and some may be more compelling than others, although it will be difficult to scale them on a common metrics. Maybe you really want to see the Coliseum, as much as you want to spend a night at Beijin opera house. You are somewhat concerned that you cannot speak Chinese. Language will also be an issue in Rome, but arguably less of one. How to decide?

Comparing the two decisions amounts to comparing two sets of arguments. Putting aside arguments that are irrelevant to the decision at hand, arguments can be positive (pros – typically, the good consequences of the decision), or negative (cons – typically, the bad consequences of the decision). Some of these arguments will be stronger or more compelling than others; however, decisions often have to be made on the basis of an ordinal ranking of the strength of the arguments, rather than on a numerical evaluation: some decision processes are of a qualitative nature. What is needed is thus a qualitative, bipolar approach to the comparison of sets of arguments. Meeting these two requirements would improve the cognitive plausibility of the method, as psychologists have argued that human choice processes are likely to be qualitative [11] and bipolar [5, 6, 15].

Ordinal ranking procedures from bipolar information have received scarce attention so far. Amgoud et al. [1] compare decisions in terms of positive and negative arguments, using a complex scheme for evaluating the strength of arguments (which possess both a level of importance and a degree of certainty, and involve criteria whose satisfaction is a matter of degree). They then compare decisions using simple optimistic or pessimistic rules, independently of the polarity of the arguments. Benferhat and Kaci [12, 3] propose to merge all positive affects into a degree of satisfaction (using the max rule). If high, this degree does not play any role and the decision is based

on negative affects (using Wald's principle, see below). If low, it is understood as a negative affect and merged with the other ones. Finally, Dubois and Fargier [8] introduce a number of qualitative, bipolar rules for comparing sets of arguments; all these rule follow the prospective work of [17] on order of magnitude calculus and thus receive an interpretation in terms of order of magnitude probabilities.

This article, which is a collaboration between a computer scientist and a psychologist, will present an extensive empirical assessment of the descriptive validity of these rules. It will not, however, thoroughly describe all their properties or include their axiomatic characterization, for which the reader can refer to [8].

2 The rules

[8] consider the simple situation where each possible decision d is assessed by a finite subset of arguments $C(d) \subseteq X$. X is the set of all possible arguments. Comparing decisions then amounts to comparing sets of arguments, i.e. subsets A, B of 2^X . X can be divided in three disjoint subsets: X^+ is the set of positive arguments, X^- the set of negative arguments, X^0 the set of indifferent ones. Any $A \subseteq X$ can likewise be partitioned: let $A^+ = A \cap X^+$, $A^- = A \cap X^-$, $A^0 = A \cap X^0$ be respectively the positive, negative and indifferent arguments of A .

Arguments can be of varying importance. In a purely qualitative, ordinal approach, the importance of arguments can be described on a totally ordered scale of magnitude $L = [0_L, 1_L]$, for example by the following function π :

$$\pi : X \mapsto L = [0_L, 1_L]$$

$\pi(x) = 0_L$ means that the decision maker is indifferent to argument x : this argument will not affect the decision process. The order of magnitude 1_L (the highest level of importance) is attached to the most compelling arguments that the decision-maker can consider. Finally, the *order of magnitude* $OM(A)$ of a set A is defined as the highest of the order of magnitude of its elements:

$$\forall A \subseteq X, OM(A) = \max_{x \in A} \pi(x)$$

2.1 The qualitative Pareto comparison \geq^{Pareto}

This rule compares the two sets of arguments as a problem of bi-criteria decision. The first criterion compares negative arguments according to Wald's [16] rule: A is better than B on the negative side iff $OM(A^-) \leq OM(B^-)$. The second criterion compares positive arguments according to the optimistic counterpart of Wald's rule:

$$A \geq^{\text{Pareto}} B \iff \begin{aligned} OM(A^+) &\geq OM(B^+) \\ OM(A^-) &\leq OM(B^-) \end{aligned}$$

¹ LTC-CNRS, 5 allées Antonio Machado 31058 Toulouse Cedex 9, France.
 E-mail: bonnefon@univ-tlse2.fr

² IRIT-CNRS, 118 Route de Narbonne 31062 Toulouse Cedex, France. E-mail: fargier@irit.fr

\geq^{Pareto} is reflexive and transitive, but incomplete. A is strictly preferred to B in two cases: $\text{OM}(A^+) \geq \text{OM}(B^+)$ and $\text{OM}(A^-) < \text{OM}(B^-)$, or $\text{OM}(A^+) > \text{OM}(B^+)$ and $\text{OM}(A^-) \leq \text{OM}(B^-)$. A and B are indifferent when $\text{OM}(A^+) = \text{OM}(B^+)$ and $\text{OM}(A^-) = \text{OM}(B^-)$. In other cases, A is not comparable with B .

2.2 The implication rule \geq^{DPoss}

This rule focuses on the most important arguments in the situation. A is at least as good as B iff, at level $\text{OM}(A \cup B)$, the presence of arguments for B is cancelled by the existence of arguments for A , and the existence of arguments against A is cancelled by the existence of arguments against B . Formally:

$$A \geq^{\text{DPoss}} B \text{ iff :}$$

$$\begin{aligned} \text{OM}(A \cup B) = \text{OM}(B^+) \Rightarrow \text{OM}(A \cup B) = \text{OM}(A^+) \\ \text{and } \text{OM}(A \cup B) = \text{OM}(A^-) \Rightarrow \text{OM}(A \cup B) = \text{OM}(B^-) \end{aligned}$$

\geq^{DPoss} is reflexive, transitive and incomplete. E.g., a set with a positive and a negative argument in 1_L is incomparable to any other set.

2.3 The ordinal bipolar rule \geq^{Poss}

This rule is simpler but less decisive. It considers any argument against A as an argument for B ; any argument for A as an argument against B ; and reciprocally. Then, the decision supported by the strongest argument(s) is preferred:

$$A \geq^{\text{Poss}} B \text{ iff :}$$

$$\max(\text{OM}(A^+), \text{OM}(B^-)) \geq \max(\text{OM}(B^+), \text{OM}(A^-))$$

This rule is complete but quasi-transitive only. That is, \succ^{Poss} is transitive but the indifference relation is not necessarily transitive. For example, when $\text{OM}(B^+) = \text{OM}(B^-)$, it is possible that indifference obtains between A and B , and between B and C as well, but not however between A and C .

Notice that \succ^{Poss} is the rule identified in [17] as a principle for order of magnitude reasoning. It is less decisive than \geq^{DPoss} in the sense that \geq^{DPoss} is a refinement of \geq^{Poss} :

$$A \succ^{\text{Poss}} B \Rightarrow A \succ^{\text{DPoss}} B.$$

2.4 Discriminating arguments: \geq^{Discri} and \geq^{DDiscri}

\geq^{Poss} and (to a lower extent) \geq^{DPoss} and \geq^{Pareto} suffer from a severe drowning effect [2, 7, 9] that is often found in purely possibilistic frameworks. For example, when B is included in A , and even if all of the proper elements of A are positive, A is not necessarily strictly preferred to B .

This drowning problem is related to the fact that the three rules \geq^{Poss} , \geq^{DPoss} , and \geq^{Pareto} do not satisfy the condition of preferential independence, a very natural principle of decision making used when the criteria (or arguments) are non-interactive:

$$\forall A, B, C \text{ such that } (A \cup B) \cap C = \emptyset : A \geq B \iff A \cup C \geq B \cup C$$

\geq^{Discri} and \geq^{DDiscri} incorporate the principle of preferential independence, by cancelling elements that appear in both sets before applying \geq^{Poss} or \geq^{DPoss} .³

$$\begin{aligned} A \geq^{\text{Discri}} B &\iff A \setminus B \geq^{\text{Poss}} B \setminus A \\ A \geq^{\text{DDiscri}} B &\iff A \setminus B \geq^{\text{DPoss}} B \setminus A \end{aligned}$$

³ A similar variant of the Pareto rule might also be proposed ($A \geq^{\text{DiscriPareto}} B \iff A \setminus B \geq^{\text{Pareto}} A \setminus B$). We did not study any further the refinements of \geq^{Pareto} , since it rapidly appeared that the rule was counter intuitive.

\geq^{Discri} is complete (as is \geq^{Poss}) but not transitive, although its strict part \succ^{Discri} is. \geq^{DDiscri} is partial (as is \geq^{DPoss}) but not transitive, although \succ^{DDiscri} is.

2.5 Cardinality rules \geq^{Bilexi} and \geq^{Lexi}

These rules are based on a levelwise comparison by cardinality. The arguments in A and B are scanned top-down, until a level is reached such that there is a difference either in the number of arguments for A and B , or in the number of arguments against A and B . At this point, the set that presents the lower number of negative arguments and the greater number of positive ones is preferred. Formally: for any level $i \in L$, let

$$\begin{aligned} A_i &= \{x \in A, \pi(x) = i\} \\ A_i^+ &= A_i \cap X^+ \\ A_i^- &= A_i \cap X^- \end{aligned} .$$

Let δ be the highest value of i s.t. $A_\delta^+ \neq |B_\delta^+|$ or $|A_\delta^-| \neq |B_\delta^-|$. Then:

$$A \geq^{\text{Bilexi}} B \iff |A_\delta^+| \geq |B_\delta^+| \text{ and } |A_\delta^-| \leq |B_\delta^-|$$

\geq^{Bilexi} is reflexive, transitive, but not complete. Indeed, if (at the decisive level) one of the set wins on the positive side, and the other on the negative side, the rule concludes to incomparability. A more decisive variant of \geq^{Bilexi} loses information about this conflict. The idea behind \geq^{Lexi} is to simplify each set before the comparison, accepting that one positive and one negative argument of A can cancel each other. In other terms, at each level j , A is assigned the score $|A_j^+| - |A_j^-|$. A top-down comparison of the scores is then performed:

$$A \geq^{\text{Lexi}} B \iff \exists i \in L \text{ such that:}$$

$$\begin{aligned} \forall j > i, \quad |A_j^+| - |A_j^-| &= |B_j^+| - |B_j^-| \\ \text{and } |A_i^+| - |A_i^-| &> |B_i^+| - |B_i^-| \end{aligned} .$$

3 Empirical Tests

3.1 Methods

How well does each rule predict the actual choices made by lay persons? To answer this question, we elaborated 33 situations of choice between two options, each option being represented as a list of positive and negative features of varying importance (see Table 1; the set of situations was designed to maximize the differentiability of the different rules, as well as to check some basic hypotheses about the qualitative nature of the decisions). Participants were 62 adults (31 men, 31 women, mean age = 24). The decisions involved ‘Poldeevian’ stamps (a fictive nation). Stamp collection provided us with a clear-cut situation of qualitative comparison. Insofar as information about the monetary value of the stamps was unavailable, they were sorted in two broad groups: the rare, coveted stamps on the one hand and the common ones on the other. This was explicitly explained to participants:

"Poldeevian stamps come in two types, **rare** and **common**. Rare stamps are difficult to find, and they are treasured by collectors. Common stamps are much easier to find, and add much less value to a collection. Among the many Poldeevian stamps, we will only be interested today in four rare and four common stamps. The rare stamps are called ARBON, BANTA, CASSA, and DIDOR. The common stamps are called WIV, XYL, YER, and ZAM."

Table 1. The 33 situations, with choices yielded by the 7 rules. Option $a^{++}(xy)^-$ has one very positive feature a and two mildly negative features x and y . \emptyset is the null option. $>$, $<$, $=$, and \sim resp. read ‘prefer option 1’, ‘prefer option 2’, ‘indifferent’, ‘options are incomparable’.

	Option1	Option2	Pareto	DPoss	Poss	DDisci	Discri	BiLexi	Lexi
1	$a^{++}(wxyz)^-$	\emptyset	\sim	\succ	\succ	\succ	\succ	\succ	\succ
2	$(wxyz)^+b^-$	\emptyset	\sim	\succ	\prec	\sim	\sim	\prec	\prec
3	$c^{++}d^{--}$	\emptyset	\sim	\sim	$=$	\sim	\sim	$=$	$=$
4	$a^{++}z^+b^{--}$	\emptyset	\sim	\sim	$=$	\sim	\sim	$=$	$=$
5	$a^{++}b^{--}z^-$	\emptyset	\sim	\sim	$=$	\sim	\sim	$=$	$=$
6	$b^{++}a^{--}$	$b^{++}(wxyz)^-$	\sim	\succ	\succ	\succ	\succ	\succ	\succ
7	$a^{++}c^{--}$	$d^{++}(wxyz)^-$	\sim	\succ	\succ	\succ	\succ	\succ	\succ
8	$a^{++}d^{--}$	$(wxyz)^+d^{--}$	\sim	\succ	\succ	\succ	\succ	\succ	\succ
9	$d^{++}c^{--}$	$(wxyz)^+a^{--}$	\sim	\succ	\succ	\succ	\succ	\succ	\succ
10	$d^{++}b^{--}$	w^+	\sim	\sim	$=$	\sim	\sim	\sim	\sim
11	w^-	$a^{++}c^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
12	$c^{++}(wxyz)^-$	$(bc)^{++}a^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
13	$d^{++}(wxyz)^-$	$(ab)^{++}c^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
14	$b^{++}(ad)^{--}$	$(wxyz)^+d^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
15	$a^{++}(cd)^{--}$	$(wxyz)^+b^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
16	a^{++}	$(wxyz)^+$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
17	b^{++}	$b^{++}z^+$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
18	c^{++}	$d^{++}z^+$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
19	$(bd)^{++}$	$(ab)^{++}w^+$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
20	$(bc)^{++}$	$d^{++}(wxyz)^+$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
21	a^{--}	$(wxyz)^-$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
22	b^{--}	$b^{--}x^-$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
23	c^{--}	$d^{--}w^-$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
24	$(bd)^{--}$	$(ab)^{--}w^-$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
25	$(bd)^{--}$	$a^{--}(wxyz)^-$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
26	$(ab)^{++}(wxyz)^-$	a^{++}	\sim	\sim	$=$	\sim	\sim	$=$	$=$
27	$(bd)^{++}(wxyz)^-$	c^{++}	\sim	\sim	$=$	\sim	\sim	$=$	$=$
28	a^{--}	$(wxyz)^+(ac)^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
29	c^{--}	$(wxyz)^+(bd)^{--}$	\sim	\sim	$=$	\sim	\sim	$=$	$=$
30	$d^{++}w^-$	d^{++}	\sim	\sim	$=$	\sim	\sim	$=$	$=$
31	$b^{++}w^-$	a^{++}	\sim	\sim	$=$	\sim	\sim	$=$	$=$
32	$c^{--}w^+$	c^{--}	\sim	\sim	$=$	\sim	\sim	$=$	$=$
33	$d^{--}w^+$	a^{--}	\sim	\sim	$=$	\sim	\sim	$=$	$=$

It is worth noting that using an abstract setting like this one, with little appeal to participant's world knowledge and little import to their personal life, is a routine procedure in experimental psychology. It allows to escape the idiosyncrasies of a given application domain (e.g., choice of an apartment, or a car), and to study a mental process under rigorously controlled conditions.

Participants' task consisted in a series of 33 choices between two options (in choices 1 to 5, the second option was the status quo). E.g., in Situation 15, participants were asked which of Club 1 or 2 they would join: Club 1 offered ARBON as a welcome gift, but required DIDOT and CASSA as a membership fee. Club 2, on the other hand, offered WIV, XYL, YER, and ZAM as a welcome gift, but required BANTA as a membership fee. Participants could choose one of four responses: (a) choose Club 1, (b) choose Club 2, (c) indifferent, one or the other, would agree to choose randomly, and (d) unable to make a decision, would not agree to choose randomly. While the third response suggests indifference between the two options, the fourth response indicates incomparability.

3.2 Results

3.2.1 Preliminaries

Choices made by participants in all 33 situations are reported in Table 2. Before we turn to the results proper, three general remarks are in order. First, a great majority of the sample (80 to 90%) used a pure qualitative ranking of the arguments, as we expected. It remains that some participants (10 to 20%) appear to go beyond a strictly qualitative evaluation of the arguments: Situations 16 and 21 (and possibly Situation 1) make it clear that these participants value four common stamps more than one rare stamp.

Second, some participants (28 to 34%) seem to re-scale negative arguments, giving them more weight than positive arguments: getting a rare stamp is less important than loosing a rare one. E.g., these 'qualitative pessimists' prefer the null option in Situation 3, or option 1 in Situation 11. Finally, some misunderstandings apparently could not be avoided, especially in the third tier of the task. E.g., a small number of participants (7 to 10%) manifest a strict preference for Option 2 in Situations 22, 23, 24, or for Option 1 in Situation 25. This rate nevertheless stays low (below 10% in any case).

3.2.2 Overall descriptive validity

As a first measure of overall descriptive validity, we computed the 'fit' of each procedure to the choices made by participants. The fit of a procedure is the *average percentage of answers it correctly predicts*, across participants. A good fit might not be a guarantee of cognitive plausibility, but a low fit is certainly an indicator of poor descriptive validity. A second measure of overall validity is what we call the 'restricted fit' of the procedure, i.e., the average percentage of answers it correctly predicts across participants, only taking into account the situations *for which it predicts a strict preference*. Table 3 reports the general and restricted fit of each rule. (We also tested the fit of the procedure suggested by Benferhat and Kaci [12, 3], which we mentioned in the Introduction – this fit was only 26%).

\geq^{DPoss} , \geq^{Pareto} and especially \geq^{Poss} fare badly in this first evaluation. The restricted fit of \geq^{Poss} may seem high, but it concerns 4 situations only (on the positive side, this shows that the principles of $>^{\text{Poss}}$ are considered as an obvious, minimal norm by the decision makers). On the contrary, \geq^{DPoss} and \geq^{Pareto} show some serious weakness with respect to restricted fit. Only \geq^{Bilexi} and \geq^{Lexi} achieve promising general

Table 2. Choices made by participants (in % of answers) in the 33 experimental Situations.

	Option1	Option2	>	<	=	~
1	$a^{++}(wxyz)^-$	0	79	21	–	–
2	$(wxyz)^+b^-$	0	–	86	7	7
3	$c^{++}d^-$	0	3	34	35	28
4	$a^{++}z^+b^-$	0	73	10	3	14
5	$a^{++}b^-z^-$	0	3	83	–	14
6	$b^{++}a^-$	$b^{++}(wxyz)^-$	7	83	3	7
7	$a^{++}c^-$	$d^{++}(wxyz)^-$	10	80	–	10
8	$a^{++}d^-$	$(wxyz)^+d^-$	83	3	3	11
9	$d^{++}c^-$	$(wxyz)^+a^-$	76	–	3	21
10	$d^{++}b^-$	w^+	14	76	7	3
11	w^-	$a^{++}c^-$	45	38	3	14
12	$c^{++}(wxyz)^-$	$(bc)^{++}a^-$	14	86	–	–
13	$d^{++}(wxyz)^-$	$(ab)^{++}c^-$	28	69	3	–
14	$b^{++}(ad)^-$	$(wxyz)^+d^-$	10	45	3	42
15	$a^{++}(cd)^-$	$(wxyz)^+b^-$	7	45	3	45
16	a^{++}	$(wxyz)^+$	90	10	–	–
17	b^{++}	$b^{++}z^+$	–	100	–	–
18	c^{++}	$d^{++}z^+$	–	100	–	–
19	$(bd)^{++}$	$(ab)^{++}w^+$	–	97	3	–
20	$(bc)^{++}$	$d^{++}(wxyz)^+$	83	17	–	–
21	a^{--}	$(wxyz)^-$	17	80	–	3
22	b^{--}	$b^{--}x^-$	73	10	7	10
23	c^{--}	$d^{--}w^-$	83	7	3	7
24	$(bd)^{--}$	$(ab)^{--}w^-$	73	10	3	14
25	$(bd)^{--}$	$a^{--}(wxyz)^-$	7	76	–	17
26	$(ab)^{++}(wxyz)^-$	a^{++}	72	28	–	–
27	$(bd)^{++}(wxyz)^-$	c^{++}	72	28	–	–
28	a^{--}	$(wxyz)^+(ac)^{--}$	90	3	–	7
29	c^{--}	$(wxyz)^+(bd)^{--}$	86	7	–	7
30	$d^{++}w^-$	d^{++}	–	97	3	–
31	$b^{++}w^-$	a^{++}	–	97	3	–
32	$c^{--}w^+$	c^{--}	90	–	3	7
33	$d^{--}w^+$	a^{--}	86	–	4	10

Table 3. Average % of answers predicted by each rule, overall or when restricted to situations when it predicts a strict preference

	Overall fit	Restricted fit
Poss	13	83
Dposs	23	57
Pareto	31	53
Discri	32	83
DDiscri	39	70
BiLexi	64	82
Lexi	77	79

fits of 64% and 77%. Although very decisive (\geq^{Lexi} is decisive in 32 of the 33 situations), these two rules also have a good restricted fit.

\geq^{Lexi} has the best fit with the choices made by any single one of the participants – it always provides the best description for all individual patterns of decisions. The second best fit is always \geq^{Bilexi} 's. The difference between \geq^{Lexi} 's and \geq^{Bilexi} 's fits is statistically reliable, as assessed by Student's $t(61) = 11.1$, $p < .001$: The probability of observing such a difference in our sample, were the fits actually similar in the general population, is less than one in a thousand. The same result holds when comparing the fit of \geq^{Bilexi} to the fits of all other procedures. From these initial considerations, only \geq^{Lexi} and \geq^{Bilexi} emerge as serious candidates for predicting participants' choices. We will now consider in more detail the case that can be made from our data in support of these two cardinality-based procedures.

3.2.3 Escaping the drowning effect by cancellation principles

Twelve situations (17–18, 22–23, 26–33) were designed so that they could elicit a drowning effect. Depending on the situation, 66 to 95% participants escape drowning. This strongly suggests that human decision-makers know their way out the classic drawback of purely possibilistic procedures like \geq^{Poss} , \geq^{DPoss} or \geq^{Pareto} .

\geq^{DDiscr} and \geq^{Discr} (on the one hand), and \geq^{Bilexi} and \geq^{Lexi} (on the other hand), represent two different ways to overcome the drowning effect. \geq^{DDiscr} and \geq^{Discr} do it by cancelling the arguments that appear in both options, thus realizing a refinement by inclusion of \geq^{DPoss} and \geq^{Poss} . The two rules \geq^{Bilexi} and \geq^{Lexi} lexi rules also allow the cancellation of positive (resp. negative) arguments of the same level, an additional refinement by cardinality (see [8] for more details).

The proportion of participants who refine by inclusion and not by cardinality is very low, less than 10 % in any case. To a couple of exceptions only, anytime a participant drowns on a situation that could be refined by cardinality, this participant had already failed to refine by inclusion on the corresponding version of the problem. Thus, in a very large majority of cases, refinement appears to operate through cardinality rather than simply inclusion.

Finally, evidence of *internal cancellation* (within a given option, a positive and a negative feature of the same level cancel each other) would count as an argument for favoring \geq^{Lexi} over \geq^{Bilexi} . In all relevant situations (3–5 and 10–15), the modal answer is *always* the one predicted by \geq^{Lexi} , and never the one predicted by \geq^{Bilexi} .

4 Conclusion

We proposed seven procedures for choosing among options represented as bipolar sets of ordinal-ranked arguments. We then asked the question of their descriptive value: How well does each of them predict the choices human decision-makers would make? For many of these procedures, the answer is: not very well, with the noticeable exception of the two cardinality rules that seem to predict human choices fairly well, and \geq^{Lexi} in particular has the highest empirical validity among our procedures.

An interesting parallel can be made in that regard between \geq^{Lexi} and the *Take the Best* decision heuristic that has been intensively studied by psychologists [4, 10, 13, 14]. When two options evaluated with respect to a series of strictly ordered criteria (in the sense that the criteria are supposed to be of very different order of magnitude so that they can be ranked lexicographically), Take the Best can be applied and amounts to choosing the option that is preferred by the most important of the criteria that makes a difference (going down

one level in case of a tie, and so forth). Applied on such a situation of lexicographically ranked criteria, the *discri* and *lexi* bipolar rules are formally equivalent to *Take the Best*. But they are able to account for other decision situations, e.g., several criteria can share the same degree of importance.

In this sense, \geq^{Lexi} is a natural extension of the widely popular (though controversial) rule advocated by psychologists.⁴ We believe that decision rules for artificial agents should be, whenever possible, as descriptively valid as they are formally sound – from the results we have reported, we see the present paper as a step towards showing that \geq^{Lexi} is blessed indeed with these two virtues.

REFERENCES

- [1] L. Amgoud, J. F. Bonnefon, and H. Prade, 'An argumentation-based approach for multiple criteria decision', in *ECSQARU'05 - LNAI 3571*, ed., L. Godo, Berlin, (2005). Springer Verlag.
- [2] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade, 'Inconsistency management and prioritized syntax-based entailment', in *IJCAI*, pp. 640–647, (1993).
- [3] S. Benferhat, D. Dubois, S. Kaci, and H. Prade, 'Bipolar possibility theory in preference modeling: Representation, fusion and optimal solutions', *International Journal on Information Fusion*, (to appear).
- [4] A. Bröder, 'Assessing the empirical validity of the "Take-The-Best" heuristic as a model of human probabilistic inference', *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **26**, 1332–1346, (2000).
- [5] J. T. Cacioppo and G. G. Berntson, 'Relationship between attitudes and evaluative space: A critical review, with emphasis on the separability of positive and negative substrates', *Psychological Bulletin*, **115**, 401–423, (1994).
- [6] J. T. Cacioppo, W. L. Gardner, and G. G. Berntson, 'The affect system has parallel and integrative processing components: Form follows function', *Journal of Personality and Social Psychology*, **76**, 839–855, (1999).
- [7] H. Fargier D. Dubois and H. Prade, 'Refinements of the maximin approach to decision-making in fuzzy environment', *Fuzzy Sets and Systems*, **81**, 103–122, (1996).
- [8] D. Dubois and H. Fargier, 'On the qualitative comparison of sets of positive and negative affects', in *ECSQARU'05 - LNAI 3571*, ed., L. Godo, pp. 305–316, Berlin, (2005). Springer-Verlag.
- [9] H. Fargier and R. Sabbadin, 'Qualitative decision under uncertainty: Back to expected utility', in *IJCAI*, pp. 303–308, (2003).
- [10] G. Gigerenzer, , and D. G. Goldstein, 'Reasoning the fast and frugal way: Models of bounded rationality', *Psychological Review*, **103**, 650–669, (1996).
- [11] G. Gigerenzer, P. M. Todd, and the ABC group, *Simple heuristics that make us smart*, Oxford University Press, 1999.
- [12] S. Kaci and H. Prade, 'Bipolar goals: a possibilistic logic characterization odif', in *Proceedings of the ECAI'04 Workshop on local computation for logics and uncertainty*, pp. 13–18, (2004).
- [13] B. R. Newell, , and D. R. Shanks, 'Take-the-best or look at the rest? Factors influencing 'one-reason' decision making', *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **29**, 53–65, (2003).
- [14] B. R. Newell, N. J. Weston, and D. R. Shanks, 'Empirical tests of a fast and frugal heuristic: Not everyone "takes-the-best"', *Organizational Behavior and Human Decision Processes*, **91**, 82–96, (2003).
- [15] P. Slovic, M. Finucane, E. Peters, and D. G. MacGregor, 'Rational actors or rational fools? implications of the affect heuristic for behavioral economics', *The Journal of Socio-Economics*, **31**, 329–342, (2002).
- [16] A. Wald, *Statistical Decision Functions*, Wiley, 1950.
- [17] N. Wilson, 'An order of magnitude calculus', in *Proceedings UAI-95*, pp. 548–555, San Francisco, CA, (1995).

⁴ While we will not push this comparison any further in the present paper, we do plan to give it due consideration in future work. In like vein, we plan to consider how other strategies identified in the psychological literature can relate to the rules we have already explored, or can be given a formal counterpart within our approach.

A Similarity and Fuzzy Logic-Based Approach to Cerebral Categorisation

Julien Erny^{1,2}, Josette Pastor¹ and Henri Prade²

Abstract. This work proposes a formal modelling of categorisation processes attempting at simulating the way information is categorised by neural populations in the human brain. The formalism mainly relies on a similarity-based approach to categorisation. It involves weighted rules that use inference and fusion techniques borrowed from fuzzy logic. The approach is illustrated by a simulation of the McGurk effect where the combination of contradictory auditory and visual stimuli creates an auditory perceptive illusion.

1 INTRODUCTION

The understanding and the prediction of the clinical outcomes of focal or degenerative cerebral lesions, as well as the assessment of rehabilitation procedures, necessitates knowing the cerebral substratum of cognitive or sensorimotor functions. This is addressed by functional neuroimaging studies. They have shown that sensorimotor or cognitive functions are the offspring of the activity of large-scale networks of anatomically connected cerebral regions [10]. However, no one-to-one correspondence between activated networks and functions can be found [11].

To understand the intricate relations between these networks and the cognitive functions, we need a model dealing explicitly with symbolic information while being strongly rooted in our knowledge of the neurobiological processes. Two complementary approaches address the problem. On the one hand, connexionist models focus on neural representation, at different scales. Biologically-plausible modelling deals with individual neurons, achieving a good level of accuracy but losing the symbolic aspect of information [26, 27]. On a coarser scale, simpler neuron models have revealed links between the neuronal substratum and simple cognitive functions [4, 13, 16, 17]. On the other hand, models from symbolic AI tend to reproduce cognitive processes without being interested in biological plausibility [3, 12, 20, 21, 24, 25]. However none of these approaches seems to answer the question of how the activation of large-scale networks follows the brain's information processing. This paper proposes an intermediary layer of interpretation, which echoes some other works [4, 6, 14]. However these works may lack a certain biological plausibility especially in the representation of categorisation processes, problem that will be addressed hereafter.

In the present paper, we first discuss the previous models and describe the requirements that a new model should address. Then we present a new approach that can fulfill some of these prerequisites. We then apply this model to phonemic discrimination. We conclude

with a discussion of the results and some perspectives.

2 EXISTING MODELLING APPROACHES

Representing information processing in large-scale cerebral networks necessitates departing from traditional connectionist methods, which tend to represent neural populations as neurons [2]. At this level, symbolic information processing is specific to the neuronal population that implements it and can be assumed to emerge from the numerical processing in the population's individual neurons. The causal connectivity approach [22] integrates this symbolic processing with more numerical aspects linked to the activation levels of cerebral regions in large-scale networks. This approach inspired probabilistic models [14] and has influenced our own work. In causal connectivity, the neuronal populations are seen as *information processors* while the axon bundles linking these populations are *information transmitters*. The information itself is represented as two-dimensioned data: i) a numeric component, called *magnitude*, stands for the overall activation of the neuronal population that processed this piece of information (thus allowing comparisons with neuroimaging data), and ii) a symbolic component, called *type*, that qualifies the pattern of the firing neurons in the population. Information is then spread in the network of populations using a dynamic Bayesian network formalism. The treatment taking place in a node can be seen as categorisation since the type of incoming information is compared to archetypes stored in the node. These archetypes represent the configuration of the synaptic weights inside the population and are modified dynamically every time information passes through, thus implementing an unsupervised, on-line learning mechanism.

However this model is lacking a primordial feature of cerebral categorisation: similarity [23]. It is known that close-by neurons will activate and fire for similar stimuli, suggesting a similarity code between stimuli based on the proximity between neurons responding to them. In the model mentioned above [14], there is no such a similarity and the archetypes are disconnected from each other. As a consequence, the model cannot interpolate between archetypes, losing the ability to compute a fine-tuned response to an ambiguous signal.

The information flowing through the brain is noisy and often uncertain. This is a major issue that has to be dealt with. Moreover, the proposed framework should have good potentials for learning on an unsupervised and online basis. Amidst classical categorisation systems in AI [19], none meets exactly these requirements. However, case-based reasoning [1, 8, 9], and fuzzy rules-based reasoning [9] provide ideas that can be a starting point for a specific formalism.

Our goal is to build a model dealing explicitly with cerebral information, while being as close as possible to neurophysiological reality, and flexible enough to incorporate new knowledge coming from

¹ U455, Inserm/Université Paul Sabatier, Pavillon riser, CHU Purpan, 31059 cedex 3, Toulouse, France; {julien.erny, josette.pastor}@toulouse.inserm.fr

² IRIT, CNRS/Université Paul Sabatier/INPT, 118 route de Narbonne, 31062 cedex 9, Toulouse, France; henri.prade@irit.fr

neurosciences. However, learning will not be addressed in this paper.

3 A MODEL OF A NODE IN A LARGE-SCALE CEREBRAL NETWORK

A node implements the information processing and propagation by a functionally coherent neuronal population. These mechanisms are based on a functional decomposition of this population.

3.1 Principle

3.1.1 Decomposition of a node

A node receives information through several inputs and emits one output. For each input, we consider a *receiving subpopulation* and, symmetrically, we consider another *emitting subpopulation*. Inside each subpopulation we consider patterns representing, i) in the case of the receiving populations, the neurons responding to a specific configuration of the input, and ii) in the case of the emitting population, the neurons firing to propagate the output. The output domain is discrete since the internal regions of the brain does not handle continuity (every continuous incoming signal, like audio waves for example, is “broken” into categories by primary sensory cortex). Receiving and emitting patterns are linked through inference rules using the following principle : an emitting pattern is linked to one receiving pattern in each receiving population. Hence, there is one inference rule per output category (i.e. emitting pattern). Although a given receiving pattern may be used in different rules, its contribution to the activation of the emitting pattern is rule-dependent. Therefore, what is referred to in the rule is not the receiving pattern *per se* (i.e. the neuron’s bodies) but a functional image of this pattern as it is used in the rule (i.e. the activated neurons : bodies *and* axons). From now on, “receiving pattern” will be used to designate those functional images (see Fig.1).

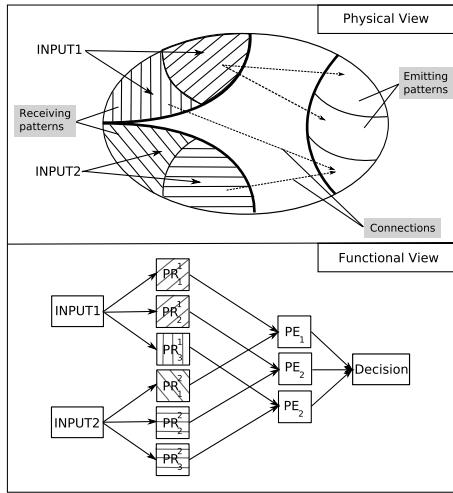


Figure 1: Physical view: the neuronal population is partitioned in two receiving subpopulations and an emitting one. Each receiving population contains two receiving patterns connected to the emitting patterns (some of the connections are not shown for readability’s sake). The functional view shows how two functional images of the same pattern can be used in two different rules.

3.1.2 Patterns

The patterns are formally fuzzy sets. In the case of emitting patterns, there exists one fuzzy set corresponding to each element of the output domain, that element being the core of the set. This element is called *referent* and characterizes the emitting pattern. The other members of the set will be referents of other emitting patterns and they belong to the set with a degree that reflects the similarity between these other patterns and the considered pattern. As for the receiving patterns, their cores contain the elements of the corresponding input domain that are perfectly recognised while the other members are the close-by and less recognised elements. The next section gives a formalism for these notions and presents the key equations describing it.

3.1.3 Inference Principle

Similarity of incoming information is computed against the receiving patterns. The more they are compatible, the more these patterns are activated. When the activity of a pattern exceeds a threshold, the pattern fires. When all the receiving patterns of an emitting pattern fire, this pattern is activated. Taking into account the similarity of the activated emitting patterns, a decision policy is then applied to decide which pattern fires the final output (see Fig. 2).

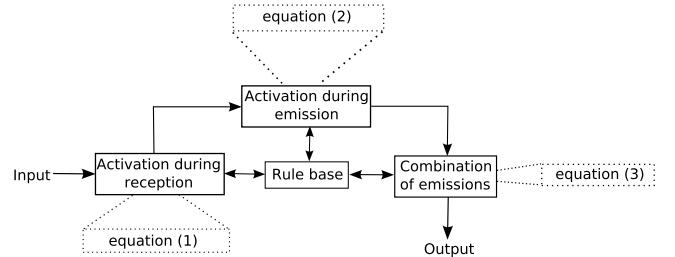


Figure 2: Inference principle

3.2 Formalism

Let X be a node with n inputs. Information is a couple (M, T) where M is the magnitude and T , the type. $M \in \mathbb{R}^+$ and T is a fuzzy set defined on \mathbb{D}_T , a discrete set. Let (M_i^{in}, T_i^{in}) be the incoming information on input i and (M_i^{out}, T_i^{out}) the output information. To keep the notations manageable, we call \mathbb{D}_i and \mathbb{D}_{out} the domains of T_i^{in} and T_i^{out} .

Let $\mathcal{BR} = \{R_i\}_{i \in [1, p]}$ be the inference rule base. We call PR_i^j the fuzzy set standing for the receiving pattern in input i for the rule R_j , and PE^j the fuzzy set standing for the emitting pattern corresponding to the conclusion of R_j . We call na_i^j (resp. sa_i^j) the level of activation (resp. the threshold of activation) of the receiving pattern PR_i^j . We call dec_i^j the firing level, i.e $\max(0, na_i^j - sa_i^j)$. Finally, we call dec^j the level of activation of the emitting pattern PE^j .

3.2.1 Activation during reception

For each time step, for each rule R_j and for each input i , the compatibility α_i^j of T_i^{in} with PR_i^j is computed as $\max_{C \in \mathbb{D}_i} \min(T_i^{in}(C), PR_i^j(C))$, which estimates the consistency

of the two fuzzy sets. Hence the activation level at time t is computed from the previous activation level by increasing it when α_i^j is sufficiently large and the pattern has not been already fired recently:

$$\begin{aligned} na_i^j(t) &= relax_X^{(1)}.na_i^j(t-1) + \\ &k.fact(\alpha_i^j, \tilde{M}_i(t)).f_{refract}(dec_i^j(t-1)) \end{aligned} \quad (1)$$

where,

- $fact$ is increasing from $[0, 1] \times [0, 1]$ on $[0, 1]$, and prevents information with a too low magnitude from passing through. \tilde{M} is a normalized magnitude ($\in [0, 1]$). $fact$ should be a threshold function, like a sigmoid for instance.
- $f_{refract}$ is decreasing from $[0, 1]$ on $[0, 1]$, with $f_{refrac}(0) = 1$ et $f_{refrac}(1) = 0$. It stands for the refractory period of the population, preventing it to activate again when it has been activated strongly before. A linear function ($f_{refrac}(dec) = 1 - dec$) or again a sigmoid can be used.
- $relax_X^{(1)} \in [0, 1]$ is called the relaxation parameter and characterises the temporal integration of the pattern. If it equals 0, there is no temporal integration.

3.2.2 Activation during emission

We have to update dec^j for each rule R_j . We assume that the combination of the emitting patterns obtained at the previous step is linear with respect to each component, hence,

$$dec^j(t) = relax_X^{(2)}.dec^j(t-1) + \sum_{i=1}^n p_i.dec_i^j(t-1) \quad (2)$$

where the p_i are parameters that have to be tuned in order to produce the more accurate simulation, and where $relax_X^{(2)}$ is the relaxation parameter for the emitting subpopulation, defined as above.

When the dec^j are known, we still have to combine and decide which pattern will fire. If there is only one emitting pattern activated ($dec^k \geq 0$), then the task is done and $T^{out} = PE^k$. However, when several rules R_{k_1}, \dots, R_{k_q} are triggered, there is a conflict that has to be solved. But first we will cumulate the activations, meaning that every time a pattern is mentioned as close to another, they will share their activations. If $rule(C)$ is the rule from which C is the referent, $dec^{rule(C)}$ is the total activation of the corresponding emitting pattern (i.e. direct activation summed with similarity activations):

$$dec^{rule(C)}(t) = \sum_{i \in \{k_1, \dots, k_q\}} PE^i(C).dec^i(t) \quad (3)$$

When all the activation levels are computed, we just choose the most activated pattern as the firing one.

3.2.3 Output magnitude

The magnitude representation is handled as causal connectivity as in [14], by a Bayesian-like propagation mechanism. The output magnitude depends on the one hand on the nature of the node (its function) and on the other hand on the level of compatibility of the incoming information. Formally,

$$M^{out}(t) = f_X^{(1)}(dec_{max}(t)).f_X^{(2)}(\overline{M^{in}(t-1)}, u) \quad (4)$$

where $\overline{M^{in}(t-1)}$ is the average over all the incoming magnitudes at $t-1$; u is a random variable standing for the noise, and dec_{max} is the activation of the firing emitting pattern. The function $f^{(1)}$ is increasing, defined from $[0, 1]$ on $[0, 1]$. Since its purpose is to generate a low output magnitude for something badly recognised, a threshold function would be a good choice. $f^{(2)}$ is a real function that may be non-linear.

3.2.4 Example

Let us consider a very simple example to make these equations somehow clearer. The focus will be on the inference mechanism letting aside the temporal aspect. To do so, we will describe the behaviour over one time step t_0 , in a completely relaxed node (there are no activation carried over from previous steps) and with all activation thresholds reduced to 0 (i.e. $\forall i, j, na_i^j(t_0-1) = dec_i^j(t_0-1) = 0$).

Consider a node with two attributes 1 and 2, with domains \mathbb{D}_1 and \mathbb{D}_2 , receiving respectively inputs T_1^{in} and T_2^{in} , with weights p_1 and p_2 . The magnitudes for each input are set to 1 and the parameter k is also set to 1. There are three “if...then” rules in the base :

$$(PR_1^1, p_1) \otimes (PR_2^1, p_2) \longrightarrow PE^1 \quad (5)$$

$$(PR_1^2, p_1) \otimes (PR_2^2, p_2) \longrightarrow PE^2 \quad (6)$$

$$(PR_1^3, p_1) \otimes (PR_2^3, p_2) \longrightarrow PE^3 \quad (7)$$

Where \otimes stands for conjunction. It then works in three steps :

- First, for each receiving pattern PR_i^j , reception activations na_i^j are computed. In this simplified case, it is just a matching rate between the pattern and the input (e.g. [9]), t_i^{in} being the input at t_0 :

$$na_i^j(t_0) = \max_{C \in \mathbb{D}_i} \min(T_i^{in}(C), PR_i^j(C)) \quad (8)$$

- Then, the direct activation dec^j of the emitting pattern PE^j is computed :

$$dec^j(t_0) = \sum_{i \in \{1, 2\}} p_i.na_i^j(t_0) \quad (9)$$

- Finally, the direct activations of the emitting patterns are combined using similarity and according to equation (3). Fuzzy sets PE^1 , PE^2 and PE^3 are shown in Fig.3. In the figure, c_1 belonging to PE^1 (whose referent is c_2) at a level h_1 means that the similarity between c_1 and c_2 is h_1 . It also means that the activation of the pattern whose referent is c_1 will contribute to the activation of PE^1 with h_1 as a discounting factor. The equation (3) gives us: $dec^{rule(c_1)}(t_0) = dec^2 + h_2 \cdot dec^1$; $dec^{rule(c_2)}(t_0) = dec^1 + h_2 \cdot dec^2 + h_1 \cdot dec^3$; $dec^{rule(c_3)}(t_0) = dec^3 + h_1 \cdot dec^1$, where $dec^{rule(c_i)}$ is the total activation of the pattern of referent c_i . The most activated pattern (i.e. for which $dec^{rule(c_i)}$ is maximum) is then chosen as the final output.

4 APPLICATION TO PHONEMIC DISCRIMINATION

As an illustration we have built with a unique node a very simple phonemic discriminator. This has permitted to reproduce an auditory illusion, the McGurck Effect.

4.1 Model

To define a phoneme, we use the articulatory theory of language that characterises a phoneme by the way we produce it. Table 1 represents

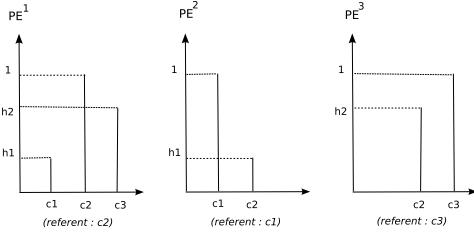


Figure 3: Emitting patterns

a simplified French phonologic system. The two described attributes (locus and mode of articulation) are our two inputs. The similarity follows what lines Table 1 suggest and we only focus on [b], [d] and [g] (i.e. [b] “is close” to [d] and “less close” to [g] and [d] “is close” to [g]).

Table 1: Consonants in the French phonologic system

locus \Rightarrow mode \Downarrow	labial	dental	velar
plosive voiceless	p	t	k
plosive	b	d	g
nasal	m	n	

The node is built with already learned categories. Assume there are four receiving patterns : $PR_{bil} = \{(bil, 1.0), (lab, 0.6)\}$, $PR_{dea} = \{(lab, 0.6), (dea, 1.0), (alv, 0.6)\}$, $PR_{vel} = \{(pal, 0.6), (vel, 1.0)\}$ and $PR_{plo} = \{(plo, 1.0), (plv, 0.6)\}$. There are three categories (i.e. emitting patterns) already implemented in the node : [b], [d], [g] and the corresponding rules are shown in expression (10). The parameters are : $k = 0.6$, and $relax^{(1)} = 0.98$ (for eq.1), and $relax^{(2)} = 0.15$ (for eq.2).

$$R_b : (PR_{bil}, 0.5) \otimes (PR_{plo}, 0.5) \rightarrow \{(b, 1.0), (d, 0.6)\} \quad (10)$$

$$R_d : (PR_{dea}, 0.5) \otimes (PR_{plo}, 0.5) \rightarrow \{(b, 0.6), (d, 1.0), (g, 0.6)\}$$

$$R_g : (PR_{vel}, 0.5) \otimes (PR_{plo}, 0.5) \rightarrow \{(d, 0.6), (g, 1.0)\}$$

Figure 4 shows the response of the node to a stimulus [b]. The stimulus is presented during 50 ms, every second, to allow neuronal processing to take place.

We see clearly that only the pattern [b] fires, the other two staying unactive all the time. Now what happens when we change the threshold?

4.2 McGurck Effect

Figure 5 shows the principle of this illusion discovered in 1976 by McGurck [18]. The visual [ga] is a person articulating a [ga] while the sound played is a [ba]. Using some recent insights about its neurophysiological nature, we implemented it using the same node but in a modulated way, i.e where the thresholds triggering the category [g] are lowered when a visual [g] is presented along with an auditory [b]. Thanks to the combination of activations in the emitting patterns, the response comes as a [d] (see Fig.6). Because of the lowering of the activation threshold, the emitting pattern [g] gets activated, and the pattern [b] is activated by the stimulus itself. If we look at Table 1, we see that between these two sounds, stands the [d]. Hence, the pattern [d] is close to the other two and gets activated when they are,

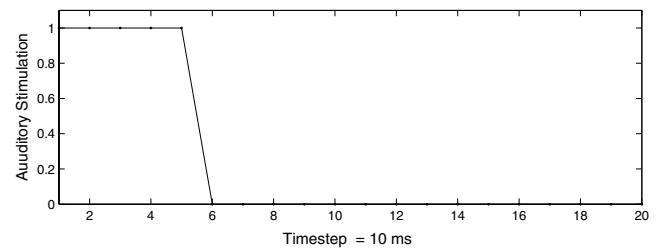


Figure 4: Activations of the emitting patterns, in response to an auditory stimulation of type [b]

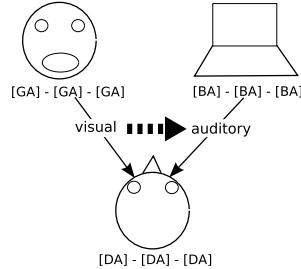


Figure 5: The simultaneous presentation of a ‘visual’ [ga] and of an ‘auditory’ [ba] leads to the auditory perception of a [da]

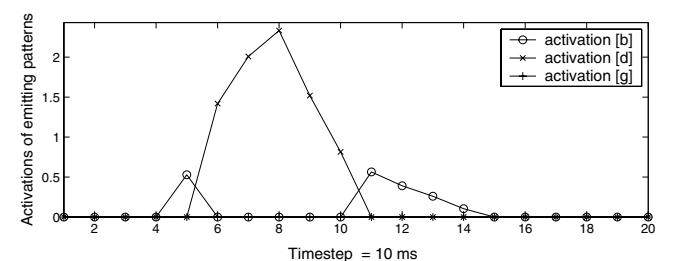
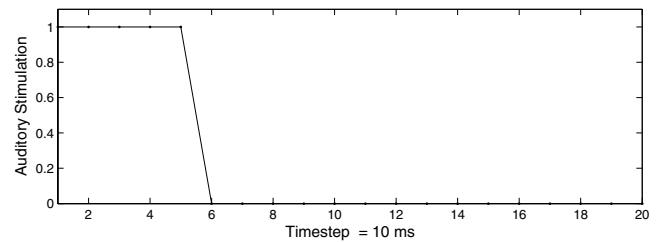


Figure 6: Activations of the emitting patterns in response to an auditory stimulation of type [b], modulated by a visual stimulation of type [g].

according to the principle of combination of activation 3.2.2. If [d] is close enough of the other two, it will exceed with this combination the activation of [b] or [g] alone and be the winner of the competition. That is what happens here.

In this simple experiment, we have shown that this model of a node can discriminate language sounds, and even reproduce an auditory illusion that was impossible to reproduce in a simple manner with previous models based on causal connectivity. However, we are not claiming that this experiment is an accurate description of how the real McGurck effect is produced by the brain. At least, it shows a demonstration of how we can use the model to simulate observable phenomena with a proper choice of parameters.

5 CONCLUSION AND FUTURE WORK

Cerebral modelling is particularly important in the medical field, to better understand the possible outcomes of a lesion or a degenerative disease. Causal connectivity [14], although promising, is still lacking a certain biological plausibility. In this work we tried to fill this gap a little, and we managed, doing that, to reproduce a phenomenon that was not modelled before by the other models of this family.

Our approach found its roots in empirical knowledge on neurophysiology and we scaled it on an “intermediary level”: the neuronal populations. This interpretation finds an interesting echo and a beginning of formal justification in the work of Balkenius and Gardenfors [5, 6], who proved that it was possible to produce non-monotonic logical inferences using activation patterns of neuronal networks. The inferences could then be encoded by weighted formulas as in possibilistic logic [7]. This calls for a comparison with the similarity-based inference mechanism as a future line of research.

Regarding the future work, dynamic learning mechanisms are being developed to build and tune the rules using only incoming information. In terms of machine learning, it is an *online* and *unsupervised* learning. We plan to test these learning abilities in two ways. They will be compared to other algorithms on machine-learning benchmarks. We will also design a simple categorisation experimental task and will compare results of the task performance by humans and by the model. Furthermore, using the work of Balkenius and Gardenfors as a starting point, we would like to formally prove that the neuronal substratum can support high level logic when one reasons in terms of patterns. This would contribute to fill the gap between connectionism and symbolic logic. Lastly, it has been shown in biology that the action of different neurotransmitters modulates the behaviour of neuronal populations (e.g. threshold lowering as in McGurck effect, see 4.2). By modelling specifically this action, we would be able to modulate our system in a systematic manner, while being even closer to biological reality [15].

REFERENCES

- [1] A. Aamodt and E. Plaza, ‘Case-based reasoning: Foundational issues, methodological variations and system approaches’, *AI Communications*, **7**(1), 39–59, (1994).
- [2] G.E. Alexander, M.R. Delong, and M.D. Crutcher, ‘Do cortical and basal ganglionic motor area use “motor programs” to control movement?’, *Behav. Brain Sci.*, **15**, 656–65, (1992).
- [3] J.R. Anderson, ‘A spreading activation theory of memory’, in *A perspective from psychology and artificial intelligence*, ed., E.E. Smith A. Collins, 137–155, Morgan Kaufmann, (1988).
- [4] M.A. Arbib, A. Billard, M. Iacoboni, and E. Oztop, ‘Synthetic brain imaging: grasping, mirror neurons and imitations’, *Neural Networks*, **13**, 975–997, (2000).
- [5] C. Balkenius, ‘Neural mechanisms for self-organization of emergent schemata, dynamical schema processing and semantic constraints satisfaction’, Technical Report 14, Lund University Cognitive Studies, (1992).
- [6] C. Balkenius and P. Gardenfors, ‘Nonmonotonic inferences in neural networks’, in *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR’91)*, pp. 32–39, (1991).
- [7] S. Benferhat, D. Dubois, and H. Prade, ‘Representing default rules in possibilistic logic’, in *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR’92)*, pp. 673–684, Cambridge, MA, (1992).
- [8] R. Bergmann and W. Wilke, ‘On the role of abstraction in case-based reasoning’, in *Proc. of the 3rd Europ. Workshop on Advances in Case-Based Reasoning*, (1996).
- [9] B. Bouchon-Meunier, D. Dubois, L. Godo, and H. Prade, ‘Fuzzy sets and possibility theory in approximate and plausible reasoning’, in *Fuzzy Sets in Approximate Reasoning and Information Systems*, eds., J. Bezdek, D. Dubois, and H. Prade, The Handbook of Fuzzy Sets, 15–190, Kluwer, Boston, Mass., (1999).
- [10] S.L. Bressler, ‘Large-scale cortical networks and cognition’, *Brain Res. Rev.*, **20**, 288–304, (1995).
- [11] J.F. Démonet, C. Price, R. Wise, and R.S. Frackowiak, ‘A pet study of cognitive strategies in normal subjects during language tasks. influence of phonetic ambiguity and sequence processing on phoneme monitoring’, *Brain*, **117**(4), 671–82, (1994).
- [12] Z. Ghahramani and D.M. Wolpert, ‘Modular decomposition in visuo-motor learning’, *Nature*, **386**, 392–395, (1997).
- [13] S. Grossberg, S. Hwang, and E. Mingolla, ‘Thalamocortical dynamics of the McCollough effect: boundary-surface alignment through perpetual learning’, *Vision Res.*, **42**, 1259–1286, (2002).
- [14] V. Labatut, J. Pastor, S. Ruff, J.F. Démonet, and P. Celsis, ‘Cerebral modeling and dynamic bayesian networks’, *Artificial Intelligence in Medicine*, **39**, 119–139, (2004).
- [15] I. Loubinoux, J. Pariente, K. Boulanouar, C. Carel, C. Manelfe, O. Rascol, P. Celsis, and F. Chollet, ‘A single dose of the serotonin agonist paroxetine enhances motor output: double-blind, placebo-controlled, fMRI study in healthy subject’, *Neuroimage*, **15**(1), 26–36, (2002).
- [16] E.D. Lumer, G.M. Edelman, and G. Tononi, ‘Neural dynamics in a model of the thalamocortical system. part 1. layers, loops, and the emergence of fast synchronous rhythms’, *Cereb. Cortex*, **7**, 207–227, (1997).
- [17] E.D. Lumer, G.M. Edelman, and G. Tononi, ‘Neural dynamics in a model of the thalamocortical system. part 2. the role of neural synchrony tested through perturbations of spike timing’, *Cereb. Cortex*, **7**, 228–236, (1997).
- [18] H. McGurk and J.W. MacDonald, ‘Hearing lips and seeing voices’, *Nature*, **264**(246–248), (1976).
- [19] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, *Machine Learning, Neural and Statistical Classification*, Prentice Hall, 1994.
- [20] M. Minsky, ‘A framework for representing knowledge’, in *The psychology of computer vision*, ed., P. Winston, 211–277, McGraw-Hill, New-York, (1975).
- [21] M. Minsky, *The Society of the Mind*, Touchstone Book, New-York, 1988.
- [22] J. Pastor, M. Lafon, L. Travè-Massuyès, J.F. Démonet, B. Doyon, and P. Celsis, ‘Information processing in large-scale cerebral networks: The causal connectivity approach’, *Biol. Cybern.*, **82**(1), 46–59, (2000).
- [23] V.M. Sloutsky, ‘The role of similarity in the development of categorization’, *TREND in Cognitive Sciences*, **7**, 246–250, (2003).
- [24] N.A Taatgen, ‘Modeling parallelization and flexibility improvements in skill acquisition: from dual tasks to complex dynamic skills’, *Cogn. Sci.*, **29**(3), (2005).
- [25] N.A Taatgen and J.R Anderson, ‘Why do children learn to say “broke”? a model of learning past tense without feedback’, *Cognition*, **86**(2), 163–204, (2002).
- [26] P.H. Tiesinga, J.M. Fellous, J.V. Jose, and T.J. Sejnowski, ‘Computational model of carbachol-induced delta, theta, and gamma oscillations in the hippocampus’, *Hippocampus*, **11**, 251–274, (2001).
- [27] X.J. Wang and G. Buzsaki, ‘Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model’, *J. Neurosci.*, **16**, 6402–6413, (1996).

Imitation of intentional behaviour

Bart Jansen¹

Abstract. In this paper a computational simulation of the imitation of intentional behaviour is presented. Important assumptions which make the problem computationally tractable are introduced and motivated. It is shown that horizontal, iterated learning, and vertical transmission schemes can be used to learn from imitation using the proposed framework.

1 Introduction

Robot imitation has been studied intensively over the past decade, as it yields the promise of extremely easily programmable robots: rather than completely specifying a task in a programming language, a task can be demonstrated to the robot. Simply by observing, the robot learns to perform the task himself. Depending on the precise application, different aspects of a behaviour must be imitated. In some cases, it is important that the demonstrated behaviour is replicated as reliably as possible (action level imitation) [11, 13, 3, 10, 5]. In goal level imitation [18, 1, 4, 7, 17], not the exact actions are copied, rather the intention of the demonstrator is deduced and a goal is achieved without reproducing the precise actions.

We propose a computational simulation of imitation of intentional behaviour. As a casestudy we investigate the imitation of spatial constraints on a blocks world and show that the proposed approach can be used in either horizontal, vertical and iterated learning schemes.

2 Case study

The experimental setup for this casestudy is depicted in figure 1². Every agent is equipped with a robot arm and can manipulate blocks on a board. With a camera system the agent can observe its own manipulations, but also the manipulations of the interacting agent. The agents can only perform very simple operations to the blocks world, like moving a block one cell to the right. A particular property of this experimental setup is that every agent acts on its own blocks world, containing the same objects. By consequence, the objects will be at different positions on each agent's blocks world, such that simple copying of actions will not result in successful imitation. Importantly, the assumption of individual environments is not essential for the presented results.

The goals the agent can learn in this blocks world express its desire to have the blocks world arranged such that it matches certain criteria. These criteria are expressed as relations over the blocks and can be combined. A possible goal is for instance $Above(A, B) \wedge LeftOf(B, C)$, supposing that the blocks world contains at least the blocks A , B and C and that the agent knows the relations $Above$



Figure 1. The agents each have a board on which configurations of blocks are demonstrated and imitated.

and $LeftOf$. This goal states that the agent pursues a configuration of the blocks world in which block A is above block B and block B is at the left of block C . Goals thus pose topological constraints on the blocks. Most often, such goals can not be reached within a single action, they thus require planning. When another agent observes such a sequence of actions, he has no unique mapping from actions to goals available. Complex processing is required to infer goals from observed sequence of actions. We propose perspective taking as a key mechanism, i.e. the imitator must be able to reason on the perceived action sequences as if he was the demonstrator.

3 Assumptions

A general strategy for deducing the intentions in others' actions can be based on the detection of structure or statistical regularities in the observed actions [6]. However, in general, the recognition of the goal of the other agent's actions is tremendously difficult. One of the reasons is that the mapping from goals to actions is very difficult to grasp. Two agents performing the same sequence of actions might pursue a different goal, while two agents performing different sequences might pursue the same goal [2]. So, the set of possible goals that an agent might pursue when performing a given sequence of actions is huge. In order to enable the agents to attribute goals to the actions they see other agents perform, this enormous search space must be constrained. This can be accomplished by taking external information into account, being for instance knowledge about human behavior in general, prior knowledge about the agent performing the actions and knowledge about the situation [2, 19, 21]. In order to restrict this search space and to allow for the efficient implementation of a goal detection method, several important assumptions are made. The first assumption poses constraints on the actions the agents can perform, the second one constrains the perceptual mechanisms and the third assumption forbids deception of the imitator by misleading actions. The fourth assumption states the existence of a memory of

¹ Department of Electrical Engineering and Informatics, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium, email: bjansen@(nospam)etro.vub.ac.be

² Picture taken from [15], kindly provided by Tony Belpaeme.

learnt goals which is actively used for the recognition of goals. Every assumption is discussed in detail below.

3.1 Action constraints

Crucial in goal recognition is the ability to recognise the actions the other agents perform as actions one can perform himself. Therefore, it is tempting to assume that all the agents have the same set of actions they can use to manipulate their blocks world. The actions we suppose to be shared are actions which manipulate the blocks world and are possibly composed of many subparts. Such an action could for instance be “move block x one cell to the left”. In order to accomplish this, the correct block should be identified, the agent should verify whether the border of the blocks world is reached, reach for the object, grasp it, and so on. We do not suppose that all agents perform all these subtasks in exactly the same manner.

3.2 Perceptual and representational constraints

In order to be able to express spatial relations on a blocks world, agent must be capable to detect those spatial relations when they observe a changing configuration of blocks on the board. Therefore, each agent is endowed with a set of relational primitives. Primitives can differ among agents, but must have the same expressive power in order to be able to imitate successfully. In [14] it is investigated how differences in expressive power affect the agent’s performance in learning goal-directed behaviour from imitation.

3.3 Planning constraints

We assume that the agents do not fool or confuse each other by performing actions which are irrelevant for the pursued goal. This assumption is based on the principle of rational action, which assumes that actions function to realize goal-states by the most efficient means available [12].

We exploit this assumption to such a degree that we suppose that the planning algorithm inside each agent is optimal. So, if the goal can be reached by performing only x actions, no $x + 1$ actions are performed. This does however not mean that all agents come with the same planning algorithm, as long as it produces optimal plans; most goals can be obtained by different optimal sequences of actions. Using this strong assumption, the imitator knows that every action performed by the demonstrator is directly relevant for obtaining its goal. This knowledge is used to seriously reduce the set of possible goals for an observed sequence of actions.

3.4 Memory constraint

The approach we propose differs from some others in that we do not allow the agent to construct a goal for every sequence of actions it observes. We use a memory-based approach; agents store goals they have encountered (their *repertoire*) and try to categorize the observed sequence of actions as a goal from this repertoire. Only when categorization fails, a new goal is constructed. Together with the memory for goals, mechanisms are implemented for maintaining it: there is a novelty detector and a mechanism to add new goals and a mechanism to remove goals which have proven to be good ones. Initially, the agent’s repertoire is empty. During the imitative interactions, the agents build up their repertoires.

Generally, the learning of a repertoire of goals and imitation of goals are considered to be two different tasks. In our approach, the agents learn a repertoire of goals and imitate goals at the same time.

4 A computational model

The computational model we introduce here is based on *imitation games* which were used in study of the emergence of vowel systems [9] and in the study of action level imitation [16]. An imitation game is an interaction between two agents which are selected randomly from a population. One of them is assigned the role of *demonstrator*, the other one acts as an *imitator*. During the interaction, the imitator tries to imitate the demonstrated behaviour and learns from it by adapting the repertoire of known goals. The computational model will be explained assuming a single teacher with a fixed repertoire of goals and a single student, starting without any goals. By observing multiple demonstrations, the imitator should acquire all goals known by the demonstrator. Further in the document, it will be shown that the framework can be used for horizontal and iterated learning as well.

Every agent A_k learns and maintains a repertoire R of goals. Each goal in the repertoire has a usage counter and a success score associated, so $R = \{(g, u, s)_i\}$. A goal g itself consists of a conjunction of predicates $g = q_1 \wedge q_2 \wedge \dots \wedge q_n$ where all predicates $q_i \in P_k$.

The imitation game is defined as follows:

1. The demonstrator randomly selects a goal g from its repertoire, builds a plan p for it and executes the plan p .
2. The imitator observes this sequence of actions and finds the best matching goal g' from its own repertoire. If no suitable goal can be found in its repertoire, the game fails and a new goal is constructed.
3. The imitator builds a plan p' for reaching the goal g' in its own environment and executes the plan p' .
4. The demonstrator observes this sequence of actions and verifies whether its initial goal g holds in the blocks world arranged by the imitator. If that is the case, the game succeeds, in all other cases it fails.
5. The demonstrator informs the imitator about the success of the game. He sends a single bit of information to the imitator.
6. The imitator adapts its repertoire.

Below, several crucial components are each described in detail.

4.1 Planning

Given the configuration of the blocks in its environment and a goal, the agent has to build a plan for reaching the goal. A plan p consists of a sequence of actions $a_1 \dots a_n$ where $a_i \in A$ and A is fixed and known to all agents in the population. Rather than finding a sequence of actions which lead to a random state in which the goal holds, the agents search for a plan which requires as few actions as possible to obtain a state in which the goal holds. Since we suppose that all agents have a planner which satisfies this constraint and since we suppose that all agents know this, the exact sequence of observed actions contains useful information for constraining the search space when inferring the goal in a sequence of observed actions. Agents know that every single action in the sequence is strictly required for obtaining the goal. Technically, the plan is built using A*-search. If no plan can be found before a given depth is reached in the search tree, the search is aborted and the imitation game fails.

4.2 Learning

The imitator starts with an empty repertoire and learns its repertoire of goals by observing and imitating the demonstrator. The imitator

uses two sources of external information to learn from: firstly, by observing the sequence of actions performed by the demonstrator, he can infer a candidate goal for the demonstrator's actions, using its own repertoire and planner. How this complex task is accomplished, is explained in detail later in this paper. If the agent fails to infer a goal, it can construct a new candidate goal based on the observed sequence of actions. Secondly, the imitator is informed about the quality of its hypothesis, since the demonstrator sends feedback about the outcome of the game.

4.2.1 Maintaining a scoring mechanism

At the end of every game (step 6), the imitator can remove goals from its repertoire which have proven to be unsuccessful in the past. Therefore, a success score is associated with every goal. When goals are first added to the agents' repertoires, their success score is set to 0.5. Once the success score of a goal drops below the parameter value ϵ , the goal is removed from the repertoire of the agent.

Whenever imitation succeeds, the success score of the used goal is increased with δ , while the success scores of all other goals in the hypothesis set³ are decreased with the same amount (success scores are limited between 0 and 1). When the game fails, the opposite is done, meaning that the score of the used goal is decreased by δ , while the scores of all other goals in the hypothesis set are increased. This implements a kind of *inhibition* [20].

4.2.2 The construction of a new goal for a given sequence of observed actions.

When goal inference fails, agents can construct a new goal, based on the observed sequence of actions. This is not an easy task since the newly constructed goal should match the goal the demonstrator of the action sequence had in mind. Since telepathy is not allowed, only the observed actions can be used, together with the knowledge deducible from the four main assumptions outlined at the beginning of the paper. The construction of the new goal consists of two major steps: first a set of goals C which meet certain criteria is determined. Later, the most plausible goal from this large set is selected.

Due to the optimality constraint, it is not possible that a candidate goal already holds in the second last state, since the last action would be useless then. Thus, at least one of the predicates which hold in the last state, but do not hold in the second last state, must be part of the goal. Therefore, the imitator determines the sets P_n and P_{n-1} . P_n contains all predicates which hold in the last state s_n . The conjunction of any of the predicates in this set could be the goal intended by the demonstrator. The set P_{n-1} contains all relations which hold in the previous state s_{n-1} . Given both sets, the imitator can find relations which became valid when the demonstrator performed the last action. He defines $X = P_n \setminus P_{n-1}$ and $Y = P_n \setminus X$. Thus, the set P_n is partitioned into two parts: X and Y . Candidate goals $c \in C$ are generated by making the conjunction of at least one element x from X and zero or more elements y from Y . All goals c share the common property that they are valid in the last state s_n but not in the previous state s_{n-1} . The goal intended by the demonstrator is thus always a member of this set.

The imitator now has to estimate how well every candidate goal c is suited for explaining the observed sequence of actions performed

by the demonstrator. In order to assess that, the imitator can imagine what actions the demonstrator should have performed if its goal was indeed the candidate c . Therefore, the imitator builds a plan for reaching the goal c starting from the initial blocks configuration of the demonstrator s_0 . However, since both agents might have a different planning algorithm, this plan can not be compared directly to the observed sequence of actions. Nevertheless, the length of the plan imagined by the imitator provides information about the goal candidate c . Since it is assumed that the planner of every agent is optimal: the demonstrator reached its goal in as few steps as possible. Thus, if the plan built by the imitator for the candidate goal c requires less steps, the candidate goal c can not be the goal the demonstrator intended. Probably, the candidate goal is too general. On the other hand, if the plan built by the demonstrator requires more steps, the goal c is probably too specific, since more specific goals tend to require more actions before they are met than more general ones. Thus, the imitator removes all goals from its candidate set C which he can not reach in the demonstrators environment in the same number of steps as the demonstrator did.

From the remaining goals in the candidate set C , the shortest goal which is not already in its repertoire is selected as the goal the imitator constructs for the observed sequence of actions. By selecting the shortest one, there is a pressure on the imitator's repertoire to be as general as possible. Therefore, it might be that the goal of the demonstrator is more specific than the goal of the imitator, which might cause the game to fail. In later interactions, more specific goals might be created in such case.

4.3 Goal categorization

When the imitator observes the demonstrator performing a sequence of actions, he has to categorize the observed sequence as one of the goals in its repertoire. This is implemented by two filters operating on the repertoire of the imitator. Initially the set C of candidate goals contains all goals in the repertoire of the imitator. Firstly, the imitator verifies for every goal in C whether the goal holds in the last state s_n of the demonstrator's actions and does not hold in state s_{n-1} . Only goals matching this initial criterion are kept as candidate goals, others are removed from C . Secondly, the imitator builds a plan for every goal c in C starting from the initial state s_0 of the demonstrator's blocks world. Thus, here the *imitator* imagines a plan the *demonstrator* could have built for a given goal and a given state. Since planners are optimal, if the initial goal of the demonstrator was indeed c , the plan imagined by the imitator must have the same length as the action sequence performed by the demonstrator. If this is not the case, it is not possible that c was the original goal of the demonstrator. Therefore only goals c with the same length as the observed sequence of actions remain in C . The remaining set is called the *hypothesis set* and from this set the goal with highest score is selected.

In several cases, the goal categorization process can fail: the repertoire of the imitator can be empty. Also after the filter operations, the candidate set C might become empty. In all those cases the game fails.

5 Experiments

5.1 Experimental setting

The environment of the agents consists of a two dimensional five-by-five simulated blocks world, consisting of the blocks $BS = \{A, B, C\}$. Agents can manipulate those blocks; a block can be moved a single cell in any of the four directions:

³ The definition of the hypothesis set is given later. For now this set can be seen as containing all goals from the agent's repertoire which match the observed sequence of actions, but which have lower scores than the candidate goal.

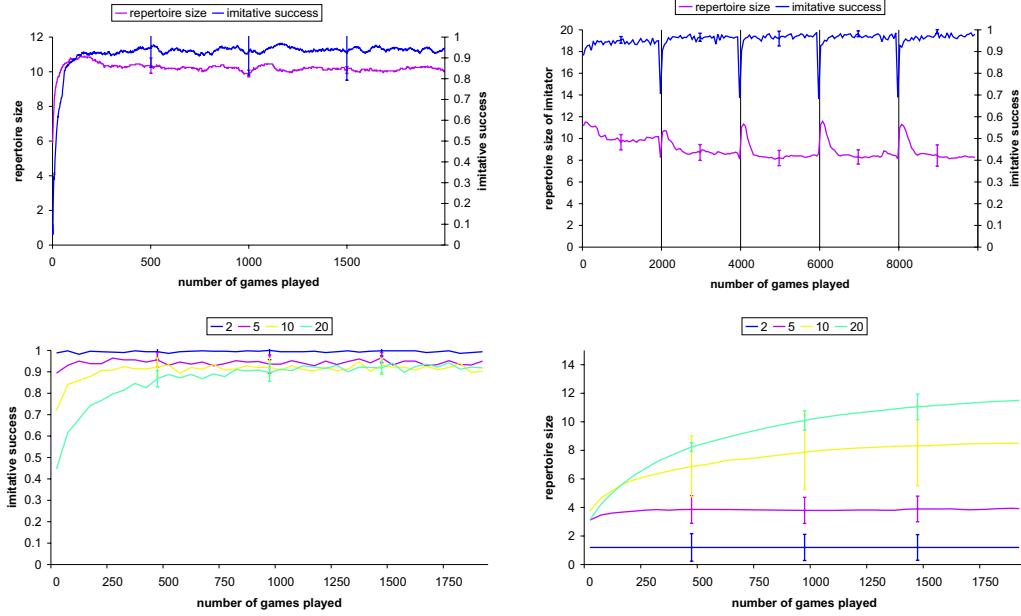


Figure 2. Top left: Results for a vertical transmission experiment. Top right: Results for an iterated learning experiment. Bottom: results for a horizontal transmission experiment.

$up(x)$, $down(x)$, $left(x)$, $right(x)$, where x denotes a block in BS . The plans built by the agents consist of several of those actions. In the experiments reported here the goals in the repertoires of the agents are represented as conjunctions of the predicates $Above?(x, y)$ and $LeftOf?(x, y)$. δ was set to 0.025, while ϵ was set to 0.1.

5.2 Measures

The performance of the agents is investigated by measuring the imitative success and the repertoire size. The imitative success is simply the fraction of games which resulted in successful imitation. A running average over a window of 100 games is calculated. Measures are explained in more detail in [15].

5.3 Vertical transmission of goals

In this first type of experiments, we consider a single demonstrator and a single imitator. The demonstrator starts with a repertoire of k different goals and tries to pass this repertoire as reliably as possible to the imitator. The goals in the repertoire of the demonstrator are generated randomly and consist of one or two predicates.

Using the parameter values stated above and $\delta = 0.025$, $\epsilon = 0.1$ and $k = 10$, imitative success is plotted in the left upper part of figure 2. Unless explicitly mentioned, all results presented in this paper are obtained by playing 2000 games and averaging over ten runs. 95% confidence intervals are plotted on the graphs whenever appropriate.

The results show that the repertoire of the demonstrator can indeed be transferred to the imitator: imitative success is stable and above ninety percent, throughout the 2000 games. The repertoire of the imitator also contains ten goals, almost from the beginning of the experiment. The goals which are learnt by the imitator are very similar to the demonstrated ones, but are not necessarily the same. Space

limitations prevent us from investigating the similarity between both repertoires in more detail. This and detailed parameter studies are presented in [14].

5.4 Iterated learning

In order to investigate the stability of the repertoire over transmission, it is useful to design an iterated learning [22] experiment. In such an experiment, an initial demonstrator is endowed with a repertoire consisting of k random goals. This repertoire is learnt by an imitator. After l interactions, the imitator becomes the demonstrator and a new imitator with an empty repertoire is created. In the top right part of figure 2 results of such an experiment are shown. The initial demonstrator starts with $k = 10$ random goals and every 2000 interactions, roles are changed. Even at the fifth generation, the repertoire of the imitator contains ten goals on average, while imitative success climbs to above 95% after an initial decline. The evolution of the repertoires over several generations is investigated in depth in [14].

5.5 Horizontal transmission of goals

In this experiment, we investigate whether a shared repertoire of goals can emerge in a population of agents, by repeatedly playing imitation games. A population consists of any number of agents, which now all start with empty repertoires and all can both take the roles of demonstrator and imitator. As all agents now start with empty repertoires, initially no games can be played, since the demonstrator also has no goals in its repertoire. Therefore, the game is slightly modified compared to the game used in the vertical transmission experiments: agents are now allowed to add a random goal to their repertoire whenever their repertoire is empty. This allows them to start the imitation games.

Results of experiments with populations of two, five, ten and twenty agents are given in the bottom part of figure 2. The graphs

indicate that stable repertoires of goals are learnt and that imitation is successful, independent of the population size.

6 Discussion

We have proposed a mechanism for the imitation of goals in a population of simulated agents. In this setup, goals are represented as spatial relations over the blocks in the agents' environments. With this approach, we deviate from the often made assumption that the goal of the agents' action is a certain configuration on the board; in our case not the configuration itself, but certain properties of the configuration are imitated.

The experimental setup was designed such that the mere copying of the actions performed by the demonstrator (*imitation of actions*) results in a failed interaction, because the agents operate on their own blocks world and thus all start with different configurations. Whether agents operate on a shared environment or each on their individual environment is not important for the model, the choice for separate environments is purely for illustrating this property.

Also the copying of the easily observable end configuration (which reduces it to *imitation of effects*) is not what is studied in the work proposed here. We and several others have argued that the endstate itself does not contain enough information for recovering the intent of the demonstrators' actions (e.g. [21, 2]). Therefore the imitator uses several sources of information for recovering the intent of the demonstrator (*imitation of goals*): the observable endstate as it was produced by the demonstrator, the sequence of actions performed by the demonstrator, context information, and a memory of previously learnt goals. Together, they allow the imitator to identify those spatial properties the demonstrator wanted to be fulfilled. The precise nature of the context information and the memory component were stated in four assumptions which were introduced and motivated.

In the approach used, the task of the imitator can not be summarized as *plan recognition*[8]. In plan recognition an agent observes a sequence of actions (a plan is operationalized) and compares this sequence of actions to every plan in its repertoire. In plan recognition, this repertoire of plans is predefined and not modified by a learning process. In this work, the agents do not store plans, but goals. Moreover, the repertoire of goals is learnt, rather than predefined.

In the vertical learning scheme, a single demonstrator starting with a predefined repertoire of goals interacts repeatedly with a single imitator, who's task is to learn the repertoire of the demonstrator. In the iterated learning scheme, the imitator can replace the teacher and the imitator is then replaced by a virgin imitator every N interactions. In the horizontal transmission scheme, a population of agents is considered. All agents start with empty repertoires and can interact with each other, both in the roles of demonstrator and imitator. Independent of the transition scheme and the precise parameter settings, we have shown both successful imitation and successful learning of repertoires of goals which are shared by all agents.

Acknowledgement

The research described in this document was performed when the author was working at the VUB AI-Lab. During his stay at the AI-Lab the author was funded by the IWT, which is the Institute for the Promotion of Innovation by Science and Technology in Flanders. Currently, the author is working at VUB ETRO and is funded by Brucare.

REFERENCES

- [1] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn, 'Towards robot cultures? - learning to imitate in a robotic arm testbed with dissimilar embodied agents', *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, **5**(1), 3–34, (2004).
- [2] Dare A. Baldwin and Jodie A. Baird, 'Discerning intentions in dynamic human action', *Trends in cognitive sciences*, **5**(4), (2001).
- [3] D.C. Bentivegna and C.G. Atkeson, 'Learning how to behave from observing others', in *Workshop on motor control in humans and robots (SAB 2002)*, Edinburgh University, (2002).
- [4] Aude Billard, Yann Epars, Sylvain Calinon, Stefan Schaal, and Gordon Cheng, 'Discovering optimal imitation strategies', *Robotics and autonomous systems*, **47**, 69–77, (2004).
- [5] Aude Billard and Maja J. Matarić, 'Learning human arm movements by imitation: Evaluation of biologically inspired connectionist architecture', *Robotics and Autonomous Systems*, (941), 1–16, (2001).
- [6] P. W. Blythe, P.M. Todd, and G.F. Miller, 'How motion reveals intention: Categorizing social interactions', in *Simple heuristics that make us smart*, 257–285, Oxford University Press, (1999).
- [7] S. Calinon, F. Guenter, and A. Billard, 'Goal-directed imitation in a humanoid robot', in *Proceedings of the IEEE Intl Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, (April 18–22 2005).
- [8] Sandra Carberry, 'Techniques for plan recognition.', *User Modeling and User-Adapted Interaction*, **11**(1–2), 31–48, (2001).
- [9] Bart de Boer, 'Self organization in vowel systems', *Journal of Phonetics*, **28**, 441–465, (2000).
- [10] Yiannis Demiris and B. Khadhouri, 'Hierarchical, attentive multiple models for execution and recognition (hammer)', in *Proceedings of the ICRA-2005 Workshop on Robot Programming by Demonstration*, Barcelona, Spain, (2005).
- [11] R. Dillman, O. Rogalla, M. Ehrenman, R.D. Zöllner, and M. Bordegoni, 'Learning robot behavior and skills based on human demonstration and advice: The machine learning paradigm', in *Proceedings of the 9th International Symposium of Robotics Research ISRR '99*, pp. 181–190, Utah, USA, (1999).
- [12] György Gergely and Gergely Csibra, 'Teleological reasoning in infancy: the naïve theory of rational action', *Trends in Cognitive Sciences*, **7**(7), 287–292, (2003).
- [13] G.M. Hayes and Yiannis. Demiris, 'A robot controller using learning by imitation.', in *Proceedings of the 2nd International Symposium on Intelligent Robotic Systems*, pp. 198–204, Grenoble, France, (1994).
- [14] Bart Jansen, *Robot imitation - The emergence of shared behaviour in a population of agents*, Ph.D. dissertation, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium, 2005.
- [15] Bart Jansen and Tony Belpaeme, 'A computational model of intention reading in imitation.', *Robotics and Autonomous Systems*, **54**(5), 394–402, (May 2006).
- [16] Bart Jansen, Bart De Vylder, Bart de Boer, and Tony Belpaeme, 'Emerging shared action categories in robotic agents through imitation', in *Proceedings of the Second International Symposium on Imitation in Animals and Artifacts.*, pp. 145–152, (2003).
- [17] Y. Kuniyoshi, M. Inaba, and H. Inoue, 'Learning by watching: Extracting reusable task knowledge from visual observation of human performance', *IEEE Transactions on robotics and automation*, **10**, 799–822, (1994).
- [18] A. Lockerd and C. Breazeal, 'Tutelage and socially guided robot learning', in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, (2004).
- [19] Andrew N. Meltzoff, 'Understanding the intentions of others: Re-enactment of intended acts by 18-month-old children', *Developmental psychology*, **31**(5), 838–850, (1995).
- [20] Michael Oiphant, 'The dilemma of Saussurean communication', *BioSystems*, **37**(1–2), 31–38, (1996).
- [21] J.R. Searle, *Minds, Brains, and Science*, Harvard University Press, 1984.
- [22] K. Smith, S. Kirby, and H. Brighton, 'Iterated learning: a framework for the emergence of language', *Artificial Life*, **9**(4), 371–386, (2003).

Dramatization Meets Narrative Presentations

Rossana Damiano^{†‡}, Vincenzo Lombardo^{†‡}, Antonio Pizzo^{*‡} and Fabrizio Nunnari^{†**} ¹

Abstract.

In recent times, information presentation has evolved towards sophisticated approaches that involve multi-modal aspects and character-based mediation.

This paper presents a novel methodology for creating information presentations based on a dramatization of the content exposition in two respects. On one side, the author plots a character's monologue that aims at achieving presentation goal and exhibits an engaging inner conflict; on the other side, the system architecture dynamically assembles the elementary units of the plot scripted by the author by implementing a tension between contrasting communicative functions.

The methodology has been applied in the implementation of a virtual guide to an historical site.

1 Introduction

A wide range of recent multimedia applications, from virtual assistants to video-games, exhibit sophisticated approaches to information presentation that involve multimodal aspects, narrative forms [13] and character-based mediation [29].

The use of artificial human-like characters improves the naturalness of the interaction with the user, making the system appear more responsive and cooperative. Much research has addressed the capability of engaging a natural face-to-face dialogue in the framework of Embodied Conversational Agents (ECA [7, 6]). The methodology for building ECAs ranges from synthesizing behavior from an abstract specification of the character to assembling pre-defined units [3, 25, 26]. In the latter case, the methodology usually involves a repository of elementary units in terms of synthetic speech, facial expressions, head movements, that are used to fill in pre-defined dialogue structures in response to user queries.

Most of the research in the field of embodied agents has focused on accounting for the emotional and expressive aspects in the multi-modal presentation of contents, while the organization of content in the exposition usually relies on simple templates derived from narrative theories like those by Propp [27] and Greimas [14]; the role of the author and the consequent production pipeline in the ECA-based system design remains unclear.

This paper focuses on methodological aspects of designing interactive information presentation systems based on artificial characters. The methodology we propose, called DramaTour, assumes that the generation of expressive behavior relies on the editing of audio-

visual elementary units in response to user input. In the DramaTour methodology, the interactional and communicative strategies of the artificial character are explicitly driven by the notion of *drama*: the presentation delivered to the user is characterized by the inner tension and the sense of direction that are typical of dramatic narrations. Information presentation becomes a dramatic monologue, in which the character exhibits an inner conflict in front of the audience, who reacts to the character's behavior. Dramatization applies to both the production of dramatic elementary units (from the writing of the script to its interpretation by the virtual character through animation) and the editing operated by the system in delivering the content to the user.

The idea of dramatizing the content, i.e. the fact that the information to be presented is encoded in a dramatic form has been recently explored, especially in the entertainment context, by the novel field of interactive storytelling. Posited at the junction of computer graphics and AI, interactive storytelling techniques aim at controlling both plot generation and real-time character behavior, mostly through planning systems [8]. Interactive storytelling involves the creation of an engaging story and its factorization into elementary units, the implementation of an AI system that reacts to the user's inputs in real time in order to assemble a dramatic performance from such elementary units, the organization of the story material within the framework provided by the AI system. These issues have been explored in some depth in the context of game design [19] and interactive drama for entertainment [22, 23].

In this paper, we apply interactive storytelling to information presentation. The working assumption is that a dramatic character, who acts in first person and shares the user's present time and space, yields a powerful effect of physical and emotional presence, especially when conveyed through an audiovisual display (cf. Esslin's notion of *dramatic media* [11]). This results in a greater effectiveness on content reception [20].

In order to create an effective system and test the practical effectiveness of the approach, the generation of the character behavior relies on pre-defined audiovisual behavior units that are assembled in real time. These units are categorized through meta-data, that serve the function of identifying their interactional and informational purposes. The applicative domain in which we are currently testing this methodology consists of guided tours in an historical site accompanied by a virtual character on mobile devices.

The structure of the paper is the following. First we describe how the notion of dramatization is put at work in this paper. Then, we present the methodology, both AI system architecture and content organization. Finally, we present the test application.

2 Dramatic issues put at work

In this section, we illustrate the aspects of the methodology that introduce the notion of dramatization in the design of a presentation

¹ [†]Dipartimento di Informatica, Università di Torino, [‡]CIRMA, Turin, Italy, ^{*}Dipartimento DAMS, Università di Torino, Turin, Italy, ^{**}Virtual Reality and Multimedia Park, Turin, Italy. Emails: {rossana,vincenzo,nunnari}@di.unito.it, antonio.pizzo@unito.it.
The authors thank Andrea Arghiento, Federica Zanghirella, Andrea De Pasquale, Cristina Galia and Federica Gilli for their help in the methodology design and implementation.

system of the type sketched above i.e. an interactive guided tour to be played by an artificial character on a mobile support in a museum, exhibition or historical site.

In line with the notion of drama formalized in [9], we see drama as the combination of two main features: the fact that drama displays action at present time and the fact that it enacts a relevant conflict related to an emotional-dramatic value concerning the characters. Drama moves toward the solution of this conflict, yielding the typical impression of movement, and does it through a sequence of elementary units, called beats [24]. Beats are pure actional units formed by a action-reaction pair.

The solution of the the conflict is called “direction”: it derives from the notion of “unity of action”, originally expressed by Aristotle [2] and clearly stated by Stanislavsky and Styan [28, 30].

2.1 Drama in Information Presentation

The principle of first-person, present-time action must be enforced by the authoring of the behavior units. The character’s behavior, in fact, is not synthesized from an abstract specification of the character, its personality, its will. So, the methodology poses some constraints on the form of data: the data encoded by the author must contain an explicit description of their informative content and of their interactional function, that the system can rely on to sequence the units according to a consistent communicative and interactional strategy.

The dialectics between different presentation modalities substantiates a dramatic conflict. The emergent behavior of the presentation system should resemble as much as possible to a carefully authored monologue, in which an internal conflict of the character is exposed to emotional response of the audience. For example, in the test application described below – a guide to a historical site – describing objects and narrating stories about the site may be put in a dialectical opposition, in which the descriptive task leaves the way, as the visit progresses, to the narration, thus realizing a shift of the character from “guide” to “storyteller”.

The advancement of drama performance, i.e., the realization of the drama direction, depends on a continuous exchange between the presentation carried out by the character and the response of the audience - intended here as the individual user - who manifests acceptance or rejection of the presentation through the input she/he provides to the system. Going back to the museum guide example, by moving to a different location, the user may implicitly communicate interest or lack of interest for the presentation, while pen-pointing on the interface controls, she/he may signal the desire to direct the presentation focus on a different object.

By applying the overall schema described above, the engagement of the audience/user is achieved by the emotional involvement in the satisfaction of the character’s goals. The characterization of emotions which the methodology implicitly refers to is the cognitive model of emotions by Ortony, Clore and Collins, in which the activation of emotions directly relates to the motivations of a rational agent [1]. The character on the virtual stage clearly wants to please the audience: as long as this goal is achieved, the character feels more and more satisfied, concretizing its initial feeling of hope into increasing self-gratification. However, this change cannot be accomplished without the passive or active intervention of the user: this fact projects the user/system interaction schema into a *meta-theatrical level*, in which the user is, at the same time, the ultimate object and an instrument of the performance.

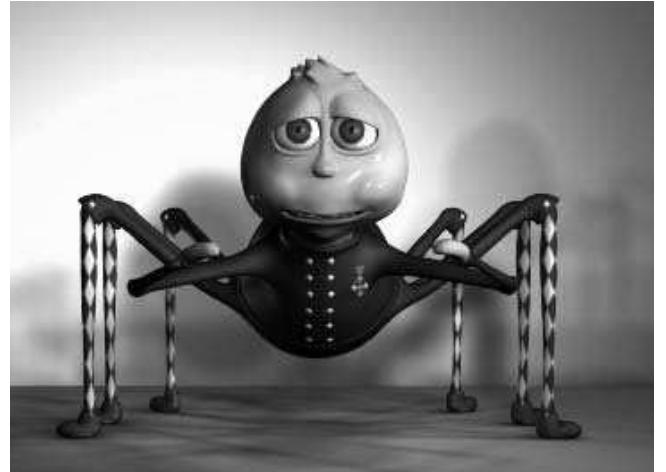


Figure 1. Carletto the spider.

2.2 An Example of Information Presentation

The test application of the methodology presented here is currently being tested in the historical location of a former residence of the Savoy family. The application consists of an interactive guided tour on a mobile device enacted by a teenage spider, which we will refer to as “Carletto”, whose family has inhabited the palace from ages. Carletto not only knows the history of the palace in detail, but knows a lot of funny anecdotes about the people who have lived there through the centuries, and is striving to tell them to the visitors.

The conflict between the role of an “audioguide”, who exposes facts orderly and plainly according to the topology of the location, and the desire to recount all the trivia and the anecdotes he knows from an historical perspective - most of which see him or his family personally involved - meet the methodology guideline of centering the presentation on an internal conflict of the character to gain the attentional and the emotional engagement of the users. Following the author guideline according to which the character itself must be carefully dramatized in the behavior units, Carletto engages in a continuous fight with the janitors, who would like to kick him out of the place.

The application is run on a mobile device. The user input consist of pen pointing on the graphical interface and localization through the use of wireless infrastructure. Abstracting from aspects of social interaction and visiting protocol, we give a short sketch of how the presentation is delivered to the user by Carletto. The visit is structured along a topological dimension, that models the palace as a set of rooms. At the beginning, Carletto follows a topological order, based on the current localization of the user. Each time the user enters a room, Carletto starts (or resumes) the presentation of the objects (furniture, artworks) in the room. When a certain amount of the room subtopics have been illustrated, Carletto happily switches to an anecdotic presentation style for a while, then gently starts inviting the user to a new room. If the user does not move, Carletto activates a “phatic function”, by playing funny games and gazing to the user from time to time.

3 The DramaTour methodology

The DramaTour methodology addresses the three issues sketched in the Introduction (story factorization, interactive story composition, drama-based and narrative-based content organization). In particular, it defines a system design that on one side provides a framework for conceptually organizing the behavior units of the system, and on the other side provides an architecture that reacts to user's inputs and assembles the units in real time. Consequently, it requires the author to create the presentation by thinking of a factorization in elementary units that the system will subsequently shape into a coherent drama direction along the interaction with the user.

The system architecture has a modular structure (see Section 3.1): the handling of the interaction with the user is mapped onto the *interaction manager*; the content organization is mapped onto the *presentation manager*; the ultimate delivery to the user in a well-edited, audiovisual continuum is handled by the *delivery manager*.

The author categorizes the units (that are scripted, interpreted and visualized) according to an ontological representation of the presentation topics and the communicative functions that contribute to the dramatization of the content delivery. The specification of the form of data (detailed in Section 3.2) concerns the set of meta data which describe the informational content conveyed by the behavior units, the interactional functions they realize and the audiovisual properties that characterize them.

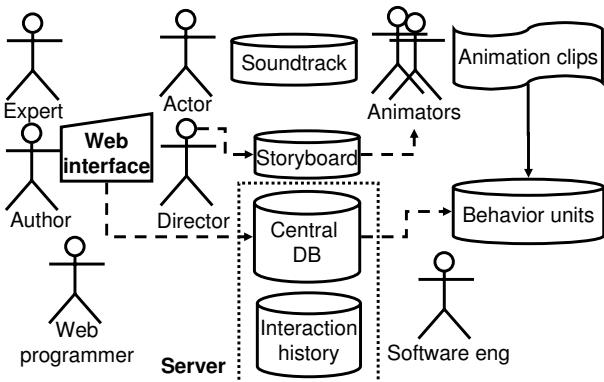


Figure 2. The production pipeline.

3.1 System architecture

The system architecture is inspired by the BDI agent model [5]: the system first selects a high-level communicative goal, then, given its library of actions, forms the intention to achieve the appropriate action; finally, it brings about the intention it is committed to by performing the action. However, the DramaTour methodology does not incorporate a full-fledged BDI model. The system does not represent goals and intentions explicitly, and does not monitor the effects of actions, as the properties of intentionality would prescribe: in fact, the context of information presentation requires only a simplified, limited interaction.

The input to the system is given by the interaction history and the user input. The system is reactive, i.e., it responds to the user input by displaying an appropriate social and communicative behavior. The system executes a *decision-execution-sensing* loop. Decision

addresses the selection of the next behavior unit; execution concerns the delivery of the unit; sensing concerns the processing of the user input.

The system views all the presentation as the realization of some communicative function [18]: in line with Grice's principle of cooperation [15], together with the presentational (informative) issues, the character must address interactional and social aspects. All of the communicative functions are hierarchically organized and, given the interaction history, some interactional functions have priority over informative functions (e.g. the character must introduce itself before providing any information).

The *informative* function is the primary task of the system, i.e., the task of providing the user with useful and relevant information during the visit. The execution of this function is assigned to the presentation manager, which handles the selection and the organization of the conveyed content through a sequence of behavior units. This module is responsible for realizing different presentation styles, according to a criterium of alternation of the presentation styles that enforces the principle of dramatization at the level of the character behavior (see next section).

The interactional functions are divided into *social interaction*, *directive* and *phatic*. Since the virtual character should qualify itself as a social agent, in order to gain believability and improve the user engagement, the system must perform some basic *social* behaviors. The *directive* function includes all the actions that the character performs in the attempt to force the user behavior in some way, like signalling conditions that may require the user to perform some action (for example, executing maintenance actions on the device on which the presentation is run, when prescribed by the visiting protocol in which the guided tour is embedded). In general, the directive function has no priority over basic social aspects (in order to enforce the notion of personification and autonomy of the virtual character) but has priority over the informative function. A relevant exception is given by the actions that, according to the visit protocol, should be executed only at the end of the presentation. The *phatic* function is activated when all other functions are applicable. Its purpose consists of signalling to the user that the character is active and willing to receive input. For example, it may be activated when the character has requested the user to perform an action of any kind - necessary for the prosecution of the interaction for maintenance reasons, and has not received any input after a given time interval.

3.2 Content organization

The behavior units, that factorize the behavior of the virtual character, constitute the knowledge base of the system. They contain multimedia content (an audiovisual clip with 3D animation and sound) and are tagged with the information that the system uses to generate the interactional and presentational behavior of the character.

The meta-data according to which the units are tagged are divided into three sets: *topic*, i.e. the description of the informative content of the unit, *communicative*, i.e. the communicative function accomplished by the unit, *editing*, i.e. the information needed for assembling the audiovisual clip with the adjacent ones. Figure 3 represents how meta-data are used by the modules of the architecture.

The *topic* section of the meta-data contains the description of the informative content of the units. The informative content is classified with respect to an ontological representation of the domain that is the object of the presentation. Topic description is necessary for the presentation manager to shape a coherent selection and exposition of the content of the presentation.

The presentation manager relies on the ontological representation of the domain information to select the content to be conveyed to the user and to structure it in a coherent way. This module follows a general strategy inspired by the focussing rules stated by Grosz and Sidner [16]. Since Grosz and Sidner's focussing heuristics have been elaborated for task-related discourse, in this methodology they have been adapted to the presentation of a set of domain facts. Task decomposition relations are mapped onto sub-topic relations, yielding the following preferences for discourse focussing:

1. Maintain focus on current topic. For example, by describing the domain according to mereological ontology, the current focus may be a piece of furniture - located somewhere in an historical location. Following a biographic description of the domain, the current focus may be an artist whose works are exhibited in a museum.
2. Move focus to a sub-topic of the current topic. With reference to the previous example, move to a subpart, a detail of the piece of furniture, or move to a certain period of the life of the artist.
3. Move focus to a the following sub-topic of the current topic. Again, the new focus may be a different detail of the previously focused detail of the piece of furniture, or a later period of the artist's life

Following sub-topic relations in an ontology according to the focussing heuristics corresponds to structuring the presentation along a certain dimension of meaning. Since the ontology is hierarchical, the focussing heuristics determine a depth-first visit of the ontology.

In principle, several meaning dimensions may be proposed to structure the same domain, corresponding to different presentation modalities. For example, the facts about an historical site may be "described" according to a topological dimension or "narrated" following a chronological dimension. In order to enforce the dramatization principle incorporated in the system design, the methodology requires the author to encode domain knowledge according to at least two different ontologies. This requirement serves the function of establishing a dialectic conflict between presentation modalities. The methodology postulates the presence of this conflict in the way the character accomplishes its presentation task, and aims at making it emerge along the interaction as a means to achieve the emotional engagement of the user.

The methodology assumes that the interaction with the user regulates the dialectic alternation between presentation modalities. So,

the methodology specifies a meta-theatrical schema according to which, as described in Section 2, user input along the interaction is interpreted as a positive or negative clue of user engagement (depending on the type user input allowed for by the specific application) and determines the presentational behavior of the character. If the user shows to dislike the current presentation modality, the character is disappointed and consequently switches to a new presentation modality. On the contrary, if the user likes the current presentation modality, the character maintains it until the related ontology has been completely explored (or explored to a sufficient degree), then switch to a different ontology.

In order to avoid abrupt transitions from presentation modalities, the domain ontologies to which the presentation refers should not be completely unrelated. For this reason, the topic of a behavior unit is expressed by encoding its position on all the available ontologies: the topic is a tuple of ontology-value pairs, where the first element of a pair refers to one of the ontologies encoded by the author, and the second element refers to a concept in that ontology.²

The system design exploits the topic description to enforce the focussing heuristics illustrated above on the set of all available ontologies: at each moment, one ontology (the reference ontology) drives the presentation, determining the active presentation modality. Each time a set of presentational units match the current topic on the reference ontology, they are ordered according to how they satisfy the focusing heuristics on the remaining ontologies, according to an author-defined preference order. In this way, when a presentation modality must be abandoned (according to the meta-theatrical schema described above, the the reference ontology changes), the transition to the new one (the secondary one) will be smooth in most cases.

The other two sets for unit categorization are the *communicative* function accomplished and the *editing* features involved.

Each behavior unit accomplishes a primitive communicative function belonging to one of the four communicative functions described above (informative, social interaction, directional phatic). It is up to the author to make sure that at least one unit matches each of the communicative functions acknowledged by the system. Moreover, each unit realizes only one communicative function. Clearly, the coordination between the system designer and the procedural author who develops the data of an application is the key to consistent scriptwriting with the system design. Moreover, it is up to the author to individuate and dramatize the character through the use of scriptwriting techniques in the authoring of behavior units.

Editing features connect some unit with another unit by interposing an audiovisual segment (called a *Transition Unit*) between them, with the aim of obtaining visual fluency [4]. The system incorporates a set of editing rules, that implement a number of editing techniques, e.g., graphic qualities (including framing, mise-en-scene, etc.) and spatial continuity. Transition units, like behavior units, are selected by the delivery manager from a repository according to the editing rules, and performs limited audiovisual adaptation if needed.

In order to assist the authoring task, a web-based authoring interface has been created to enter the application data (behavior units and transition units) and to define the meta-data according to which data are classified by the system (topics, communicative functions and editing features).

The system assumes that behavior units are self-contained, i.e. that

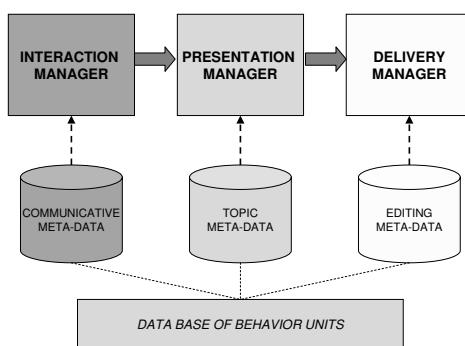


Figure 3. The system architecture according to the DramaTour methodology

² If the topic of a behavior unit is not present in one of the domain ontologies, the value element for that ontology will refer to the root of the ontology, meaning that it may equivalently subsume any specific topic in that ontology.

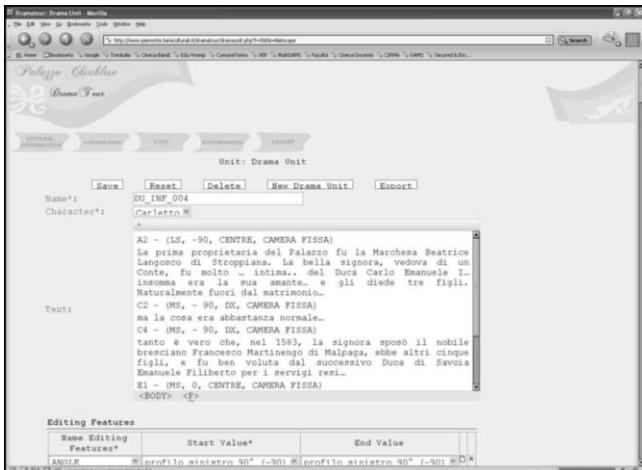


Figure 4. Authoring web-based interface

each of them accomplishes the execution of some specific communicative function, or, if its function is tagged as presentational, that it conveys a unit of meaning according the ontological representation.

3.3 Implementation details

The current implementation is based on common hardware available on the consumer market and mostly on open-source software. The visit server, that follows the specifications described in Section 3.1, is implemented in Java (<http://java.sun.com>), while the data base system is mySQL (<http://www.mysql.com/>). The web-based authoring interface has been developed in PHP (<http://www.php.net/>).

The client is written in Java and runs on an ASUS A636 PDA (PocketPC series). The video clips implementing the behavior units are encoded in Macromedia Shockwave Flash (<http://www.macromedia.com>). The client pilots the media player by sending text-based commands to localhost. The PDA also supports a localization client, that provides the user's current location, i.e., it identifies the room in which the user is currently situated.

Since the DramaTour methodology is media-independent, beside the PDA-based version of the virtual tour, a web-based guided virtual tour of the same location has been developed by using the same visit server. The web interface simulates the tour in the virtual space, by proposing to the user a PMVR (QuickTime VR) representation of each room, accompanied by the sequence of clips in which Carletto provides information about the room.

4 Conclusions

The DramaTour methodology presented in this paper is modeled on the typical workflow required by the production of a semi-automatic character-based presentation. Such a simplification of the process of system design and on-the-fly multimedia generation poses some expressiveness limitations to the author. However, we believe that the task assignment devised by the methodology between the system designer and the author represents a reasonable trade-off. Author's scripting is guided by a set of well-defined constraints on data content and form which are functional to the needs of the system. System design guidelines, in turn, enforce the key notions of character, storytelling and dramatization in an explicit way, by posing the method-

ology half-way between fully intelligent, experimental systems and off-the-shelf scripted systems for practical applications.

REFERENCES

- [1] G. Clore A. Ortony and A. Collins, *The Cognitive Structure of Emotions*, Cambridge University Press, 1988.
- [2] Aristotle, *On the Art of Poetry. I. Bywater (transl.)* 1920, Clarendon, Oxford.
- [3] N. O. Bernsen and L. Dybkjaer, 'Domain-oriented conversation with h.c. andersen', in *Proceedings of ADS 2004*, pp. 142–153, (2004).
- [4] F. Biral, V. Lombardo, R. Damiano, and A. Pizzo, 'Cyrano goes to hollywood: a drama-based metaphor for information presentation', in *AIMS 2003*, pp. 17–24, (2003).
- [5] M.E. Bratman, *Intention, Plans, and Practical Reason*, Harvard University Press, Cambridge (MA), 1987.
- [6] A. Bryan Loyall and Joseph Bates, 'Personality-rich believable agents that use language', in *Proc. of the First International Conference on Autonomous Agents*, ed., W. Lewis Johnson, pp. 106–113, (1997).
- [7] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill (eds.), *Embodied Conversational Agents*, The MIT Press, Cambridge, Massachusetts, 2000.
- [8] M. Cavazza, F. Charles, and S.J. Mead, 'Interacting with virtual characters in interactive storytelling', in *Proc. of the First Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, (2002).
- [9] R. Damiano, V. Lombardo, and A. Pizzo, 'Formal encoding of drama ontology', in *LNCS 3805, Proc. of Virtual Storytelling 05*, pp. 95–104, (2005).
- [10] L. Egri, *The Art of Dramatic Writing*, Simon and Schuster, New York, 1960 (1946).
- [11] M. Esslin, *The Field of Drama*, Methuen, London, 1988 (1987).
- [12] S. Field, *The definitive guide to screen writing*, Ebury Press, London, 2003.
- [13] N. Gershon and W. Page, 'What storytelling can do for information visualization', *Communications of the ACM*, **44**(8), 31–37, (2001).
- [14] A. J. Greimas, *On meaning: selected writings in semiotic theory*, Pinter, London, 1970.
- [15] H. P. Grice, 'Logic and conversation', in *Syntax and Semantics*, eds., P. Cole and J. L. Morgan, volume 3, *Speech Acts*, 41–58, Academic Press, New York, (1975).
- [16] B. J. Grosz and C. L. Sidner, 'Attention, intentions, and the structure of discourse', *Computational Linguistics*, **12**, 175–204, (1986).
- [17] J. Hatcher, *The Art and Craft of Playwriting*, Story Press, Cincinnati, Ohio, 1996.
- [18] R. Jacobson, 'The speech event and the functions of language', *On Language*, (1990).
- [19] H. Jenkins, 'Game design as narrative architecture', MIT Press, (2004).
- [20] B. Laurel, *Computer as Theater*, Addison Wesley Longman, Reading, MA, 1993.
- [21] Y. Lavandier, *La dramaturgie*, Le clown et l'enfant, Cergy, 1994.
- [22] M. Mateas and A. Stern, 'Integrating plot, character and natural language processing in the interactive drama faade', in *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment TIDSE 03*, (2003).
- [23] M. Mateas and A. Stern, 'Structuring content in the faade interactive drama architecture', in *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, (2005).
- [24] R. McKee, *Story*, Harper Collins, New York, 1997.
- [25] A. Nijholt and D. Heylen, 'Multimodal communication in inhabited virtual environments.', *International Journal of Speech Technology*, 343–354, (2002).
- [26] I. Poggi, C. Pelachaud, F. de Rosis, V. Carofiglio, and B. De Carolis, 'Greta. a believable embodied conversational agent', *Multimodal Intelligent Information Presentation*, (2005).
- [27] V. Propp, *Morphology of the Folktale*, University of Texas Press, 1968.
- [28] K. S. Stanislawski, *An Actor Prepares*, Eyre Methuen, London, 1980 (1936).
- [29] O. Stock and M. Zancanaro (eds.), *Multimodal Intelligent Information Presentation*, Springer, 2005.
- [30] J. L. Styan, *The Elements of Drama*, University Press, Cambridge, 1963(1962).

MAMA: An architecture for interactive musical agents

David Murray-Rust¹ and Alan Smaill² and Michael Edwards³

Abstract. In this paper, we present MAMA — an architecture for interactive musical agents. This system uses a theory of Musical Acts, based on Speech Act Theory to support the agents interaction. We discuss the basics of a representation language which these agents can use to represent and reason about music. We present a case study system based on these ideas, and discuss its ability to support distributed execution of a minimalist score.⁴

1 INTRODUCTION

Agent based music offers an exiting range of possibilities for the composition and performance of music. The distributable and mobile nature of agents suggests possibilities for remote collaborations and truly global music; the potential intelligence of agents allows for complex responses, understanding of high level inputs and a combination of flexibility and organisation; and the “human-like” properties of agents give us a way to explore models of human interaction through groups of musical peers working with dynamic material.

The “Live Algorithms for Music” group⁵ defines some characteristics that a software system needs in order to be considered a live algorithm, which are designed to coincide with it being an interesting musical partner. A Live Algorithm must be able to interact musically without any human intervention; it must make creative contributions to the music which is being produced; it should avoid rule based approaches which give a simplistic mapping between input and output. [4, 5] give an example of a particular group of live algorithms — the use of a swarm to create musical output. These systems tend to avoid traditional AI approaches of reasoning and planning, and concentrate on a lower level engagement with the musical surface. Another example of a live algorithm is the Continuator [15], which uses hidden Markov models to learn a player’s style, and respond with more music in the same vein.

Recently, there have been many projects aimed at creating “net” music - music which allows realtime collaboration over some form of network, whether a local area network, or the internet⁶. These projects have been categorised in [18] into four distinct approaches:

Server A server is used to send musical data to disconnected participants, but no interconnection or communication takes place between players

¹ School of Informatics, University of Edinburgh, UK, D.S.Murray-Rust@sms.ed.ac.uk

² School of Informatics, University of Edinburgh, UK, smaill@inf.ed.ac.uk

³ Department of Music, University of Edinburgh, UK, michael.edwards@ed.ac.uk

⁴ Audio recordings of the system are available at <http://homepages.inf.ed.ac.uk/s0239182/MAMA>

⁵<http://www.livealgorithms.org>

⁶For a more exhaustive survey, see [1].

Bridge The network is used to create a bridge to facilitate traditional interaction as if distanced players were in the same place

Shaper A central system generates musical material, which the participants collaboratively shape

Construction Kit Creates a multi-user composition environment, where participants can manipulate and shape each other’s music

Two systems are of particular interest here. Firstly the CODES system [12], which enables participants to cooperate in the construction of a “musical prototype” of a piece of music. “Lines” are made by choosing one of a set of patterns at each stage, with users able to create and modify patterns. The use of patterns allows untrained musicians to join in with the prototyping.

Secondly, the PIWeCS system [19] is a marriage of intelligent agents, electroacoustic processing and network music. A variety of traditional Maori instruments were played, and loaded into MAX/MSP (a graphical dataflow language, with extensions for sound processing). Users can then control parameters of these samples via a web interface, to create an arrangement. To allow for users with a low skill level, agents would listen to the output of the users, and supply additional material, or enter into dialogue as appropriate.

1.1 Background

This work is influenced by the characterisation of different forms of interaction between agents based on the notion of *communicative act*, which in turn draws upon the notion of *speech act* from linguistic pragmatics. The motivation behind the use of communicative acts to structure agent interaction is that the different communicative acts, such as “agree”, “cancel”, “call for proposal”, call for different forms of interaction; agreement on a set of such acts enables the reuse of composite communicative acts, and a platform for reasoning about such interactions, as well as supporting interoperability (see for example [7, 20]). This approach also naturally invokes the notion of the *intentionality* of the agents, and the opportunity for agents to reason about the intentions of other agents in the course of communication [9].

We exploit the analogy between this notion of intention, and that of musical intentionality present when musicians improvise and perform together; such musical interaction is often described in terms of “conversation” or “discussion” between the musicians, suggesting further the relevance of work on dialogue in the natural language case. The musical case differs in some respects, however.

First, musicians typically perform at the same time as other musicians, rather than taking turns as in natural language dialogues; the architecture below reflects this, in that the outputting of musical statements cannot be taken as an atomic step, but takes place in real time while monitoring simultaneous musical activity of other agents. Second, it is important to model more than interaction between two

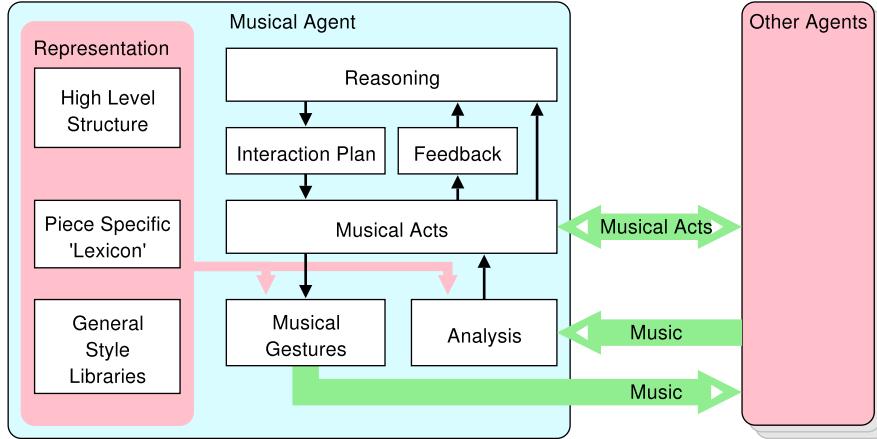


Figure 1. Musical Agent System Architecture

agents, which is the case usually treated in dialogue, with some exceptions ([2, 17]). Finally, we make no use of the notion of “literal meaning” of a musical statement that might correspond to the literal meaning of a natural language statement.

There have also been a few more direct approaches to the use of multi-agent systems in music. Dixon [6] uses a set of agents to track beats in audio streams. [21] details the construction of a musical multi-agent system where a group of agents adjust and play patterns in realtime, with themselves and human musicians. [16] describes an open architecture for Musical Agents, based on Java.

1.2 Paper Structure

In this paper we present our work on the MAMA (Musical Acts - Musical Agents) architecture. We describe the architecture, and give an overview of musical acts. We then discuss some issues of representing music for agents, and conclude by describing a current case study.

The overall aim of our work is to create a society of agents which can improvise music with each other, and interact with humans (either musically or through a higher level interface). These agents will use theory of Musical Acts, analogous to Speech Acts, to inform their music making (building on work in [13]. This is different to current approaches for a number of reasons:

- We focus on the logical aspects of computational music making, rather than the expressive and reactive aspects.
- We focus on the interaction between agents, rather than the generative capabilities of any particular agent.

2 Musical Agents Architecture

By creating a system of musical agents, we aim for a point between computationally assisted composition and interactive musical systems. The agents use high level representations of a piece of music to both inform their interaction with other agents and generate musical output. This is aimed at a range of situations where individual performers have some degree of autonomy, but they are working within a structure. Benson [3, 26-30] gives a series of increasingly “free” types of improvisation on a score. We are looking primarily at:

- I₁ The players fill in certain details which are not specified by the score
- I₂ The players add notes to the score, in a manner expected by the composer (e.g. trills)
- I_{3,5} The players add whole measures, sections etc. to the score; if this is in a manner approved or expected by the conductor, it is (I₃), otherwise, it is (I₅).
- I_{7,8} The score is changed considerably, reharmonisation, melodic alteration etc. The difference between I₇ and I₈ is that of recognisability - in the former, there is some obvious connection between the score and the rendition, but in the latter there is not.

The musical agent system is intended as a layer of “Musical Middleware” for composers and musicians (Figure 1); to create a piece, the composer:

- specifies some aspects of the high level structure
- provides a lexicon - a set of fragments and constraints on their usage - to the system.
- provides a set of general style libraries for the particular style(s) of music being created.

The agent system will maintain a high level representation based on this score. By reasoning about the current state of the score, the actions of other agents, and its own internal state, the agent will be able to construct an interaction plan, to guide interaction with other agents. This may suggest to the agent different general roles to take (such as soloing or keeping a groove), and it may result in specific courses of action - such as playing a response in a call and response section.

From the interaction plan, the agent constructs a specific set of musical acts to execute. These acts (see next section) cover the introduction, assessment and development of different musical ideas, and provide the substrate on which interaction is based.

A musical act may well only define a small section of the musical surface, and there may well be times when an agent is not executing any acts at all - for example, when supporting a soloist, one’s playing would be relatively free of new ideas in order that the soloist’s voice is more clearly heard (much as one is silent in conversation to allow another party to speak). This means that musical acts are a supplement to the generation process, and the bulk of the musical

surface may have to be generated from other sources. We hence use a combination of the high level representation with both style- and piece-specific information.

As the agent generates output, it also listens to the output of others. This output is analysed according to whatever features the particular agent knows about. From this analysis, and from the musical surface itself, the agent extracts the musical acts which other agents have performed. The activities of the other agents are used in the reasoning process, and also to determine feedback about the agent's actions.

It can be seen from Figure 1 that each agent has two communication channels to the rest of the world. Firstly, there is the musical channel, where musical information, in some form, is transferred between agents. Secondly, there is a formal channel, which allows the direct communication of Musical Acts. This could be used to transmit representations of the hand signals used in samba drumming, a nod to indicate that one agent likes another's contribution, or other traditional extra-musical gestures. It also provides the possibility for a higher level interaction with the agents - a user interface could deal directly in Musical Acts, giving a way to organise the playing of ensembles, without having to define low level detail, and while keeping the autonomy of the individual agents intact. Finally this channel provides a means of injecting, after suitable processing, data from other modalities, such as the sensor data in an interactive installation.

In order for an agent to reason effectively, it must receive feedback about the quality of its actions. We have three types of feedback that an agent is interested in:

- Satisfaction of internal goals: these may be given by the composer, or arise during the performance of a piece.
- Direct feedback from other participants; the equivalent of a smile or nod from another band member could be given by a human participant directly through some interface.
- Interaction feedback; when someone takes up an idea that an agent suggests, this results in positive feedback, even more so if they develop the idea, or enter into dialogue about it.

2.1 Interactions

To support a rich level of interactions, we will use a system of Musical Acts [13], which are analogous to Speech Acts as used in common agent communication languages (e.g. FIPA-ACL [8]). This is built on a set of modal operators as follows:

$BM_i p$ (BelMusic) “Agent i believes p to be a part of the shared representation of all agents”.

$UM_i p$ (UndefinedMusic) means “Agent i does not believe there is a commonly accepted value for the part of the representation covered by p ”.

$IM_i p$ (IntendsMusic) means “Agent i intends p to become part of the shared representation”.

$PM_i p$ (PersonalMusic) means “Agent i has p as part of its private representation”.

There is also the idea of musical “compatibility”. A musical proposition ψ is compatible with ϕ ($Comp(\psi, \phi)$) if ψ is in some way an extension of ϕ . The exact details of compatibility may be defined differently for different features, but examples would be extending or ornamenting a melody (compatible), adding degrees to a chord (compatible), changing one of the degrees in a chord (incompatible).

By combining the possible states of the world, and the values of the modal operators, the following set of musical acts may be obtained (also shown in Table 1):

Inform If an agent is expressing a facet of the musical structure which it believes has not been explored - it has no value in its public representation - then it simply INFORMS the other agents of it's desire that the representation be modified thus.

Confirm If agent A INFORMS of some facet for which agent B has no value, and B is content with the value A has suggested, then B can CONFIRM As suggestion

Disconfirm As above, except B is not happy with A's value, and DISCONFIRMs it. This could take the form of B playing something which is not compatible with A's exposition

Extend If A has a value in its public representation which it wishes to alter in a manner which is compatible with the current value, it can EXTEND the current value.

Alter If A has a value in its public representation which it wishes to alter in a manner which is *not* compatible with the current value, it can ALTER the current value.

Preconditions			Act
Agent i	$B_{i,j}$	Other	
$UM_i \phi \wedge IM_i \phi$	*	-	Inform-Music
$IM_i \phi$	$IM_j \phi$	-	Confirm-Music
$\neg IM_i \phi$	$IM_j \phi$	-	Disconfirm-Music
$BM_i \phi \wedge IM_i \psi$	*	$Comp(\phi, \psi)$	Extend-Music
$BM_i \phi \wedge IM_i \psi$	*	$\neg Comp(\phi, \psi)$	Alter-Music

Table 1. Musical Acts and their preconditions

2.2 Music Representation for Agents

We had several design goals when creating the music representation language. In particular:

- It would be human readable, to allow composers to understand their own and others' scores.
- It would not be tied to traditional Western music, and potentially represent as many types of music as possible
- It would support pieces with dynamic structures; sections of music could be re-ordered, extended and omitted at run-time.

We took several inspirations for our representation; in particular, this is a development of work carried out in [14]. We are working towards a target set of pieces which can be represented. To get a well balanced set of features, we are looking at:

Latin The “Latin Real” fake books provide a high level representation of Latin pieces; they specify structure and some musical features, with the players left to fill in the details using their own understanding of the style of music

Minimalist By looking at In C we examine a piece where the notes are given, but the structure is dynamic, determined by the decisions of the agents.

By creating a system which can represent these musics we ensure sure that it can be extended to deal with a wide range of music.

2.2.1 Structure

The hierarchical decomposition of music is not new, and has been widely used in many computational representations of music - [10] is a recent implementation of Lerdhal and Jackendoff's Generative Theory of Tonal Music [11].

In our system, a piece of music is divided up into a hierarchical tree of “Sections”. The leaf sections are used to define aspects of

the music to be played, while non-leaf sections are used to order the leaves. Attributes are inherited down the hierarchy, making it easy for a composer to create many similar sections, and specific relations may also be given.

Sections have several controls over their ordering. By default, sections would be played out in the order they are given. It is possible to specify different orderings, so that a section may use a decision rule to choose which of its subordinates is played next.

In order to exchange information about music, the agents need to have a way to refer to sections of the music. We use path like structures here, so that the exact place a section is used can be specified. This allows for agents to tell each other what section they are currently playing, and propose changes to certain sections of the piece.

Many scores supplement their notation with a set of playing directions. For certain styles of music, it is essential that we can represent these directions such that agents can use them - in some minimalist and contemporary works⁷ the directions define a large part of the structure of the piece.

We represent these directives as behaviours which an agent may enact, and specify them as a combination of a condition and an action; the agent is responsible for maintaining certain variables, to do with the musical surface and its own internal state, which can then be used in an expression which is the condition of a directive. The action of a directive can have a structural effect, such as choosing to move from one section to the next; it can trigger the use of certain musical material, such as playing the response to a call; it can shape output which has already been generated, such as adjusting dynamics to create a crescendo.

2.3 System Infrastructure

In order to avoid having to deal with realtime issues at many levels, while maintaining realtime output, we have made the following decisions and simplifications:

- There is a central “Conductor” in the agent system, who is responsible for gathering the output of all the agents and translating it into MIDI or audio data.
- A piece occurs at a certain tempo, determined by the conductor. Individual agent produce their output at times specified in beats.
- An agent creates output in a buffer, which is then periodically sent to the conductor for output. The conductor will ask for a section of output covering a given time span.

This means that an agent’s only temporal responsibility is to ensure that the playahead buffer is filled, and to return a requested chunk of it in a timely fashion when asked.

The information available to an agent about the playing of other agents is always one time span behind. There is hence a tradeoff between reactivity of the agent system, and the latency of the network (and message passing overhead).

The agents are situated within a virtual space, and can be moved around by means of a graphical interface. Their positions within the space affects where their sound appears in the stereo field, and can be used to alter how much attention is paid to the output of other agents.

3 CASE STUDY: In C

We chose “In C”, by Terry Riley both due to its impact at the beginning of the Minimalist movement, and because of its suitability

to agent based techniques. The score consists of a series of short fragments of music, ranging from half a beat to several bars. Each performer plays the fragments of music in turn, repeating each indefinitely before moving on to the next. Directions are given such as:

- performers should stay within 2 or 3 patterns of each other
- performers must listen carefully, and should occasionally drop out and listen
- a pattern may be played in unison or canonically in any alignment with itself
- the group should aim to merge into a unison at least once or twice during the performance

To implement this, in our agent system, we provided the following components:

Representation We created a score which details the notes provided in each section. We then added a tacet section which agents can jump to to take a break, and special section for the end - once all players have arrived on the final section, the group crescendos and diminuendos a few times, with players dropping out on each diminuendo.

Musical Acts When an agent begins a new section, it sends a message to all the other agents to inform them of the section it is starting. This is done through the formal channel (see Figure 1)

Reasoning In order to follow the direction that players should stay within about 3 sections of each other, each agent tracks the progress of the others through the piece. The relative position in the score of the other agents is perceptually weighted (scaled by the inverse square distance and the volume) and summed to give a total weighting for the relative progress of the group. We then use a sigmoid function to turn this into a probability, with control over both the base probability of moving from section to section, and how much spread the agents are comfortable with.

Analysis Some low level features such as dynamics are analysed, and some more complex features such as syncopation and unity are planned. This will feed into the analysis section, and allow agents to make changes in their output to increase or decrease syncopation, for example.

4 RESULTS

We had a variety of aims in implementing this piece. Firstly, we were testing whether the system infrastructure could deal with a reasonable number of musical agents playing together, without dropout. Secondly, we wanted to test that by using a set of simple directions, we could create a result which was musically pleasing, and demonstrated some behaviours we would expect from a human ensemble.

We found that the system could support a useful number of simple agents - so far In C has been played by 20 musicians on a standard PC, using the internal synthesis engine provided by Java. We found that the system is quite sensitive to other applications running on the computer, although CPU usage remains reasonably low. Use of an external synthesiser is expected to improve timing and quality of sounds produced.

We have not performed a serious experiment concerning the musicality of the current system, as we do not feel it is warranted at this stage. The fact that its musical output is generally pleasing is a function of the piece in question as well as the agent system itself.

⁷e.g. Stimmung (Stockhausen), In C (Riley), Canto Ostinato (ten Holt)

5 DISCUSSION AND FUTURE WORK

In this paper we have presented an architecture for interactive musical agents, and discussed a prototype system which implements some of this architecture. This is useful for a proof of concept, but there are many possibilities for further extension.

There are several comparable systems at present, but they all address slightly different concerns: Pachet's Continuator [15] is capable of high quality interaction with human musicians, but lacks higher level descriptions of music; the Andante project [16] gives an architecture for mobile musical agents, but focuses on the low level aspects of this area; and Wulfhorst's system [21] lacks high level structure.

It is clear that the use of a formal set of musical acts allows some very precise communication between the agents, and it is certainly less computationally intensive than having every agent analyse the output of every other agent. This makes the prospect of hybrid human/machine interaction an interesting area, as humans cannot easily give this privileged view onto their mental states. It should be possible to analyse human playing, and pick out musical acts, which could then be distributed to the agents; it may also prove interesting to supplement a traditional instrument with some means of generating metadata about one's playing - e.g. the capture of physical gestures, or a specific physical interface.

Often with agent systems, there is a question of autonomy, about how different this approach is to standard object oriented techniques, or even distributed OOP. This has a sister question in musical circles, concerning the autonomy of a player within a musical ensemble. In certain styles of music, the musicians are given highly precise descriptions of what to play, whereas in others the players have a lot of creative control over what they play. In terms of Benson's levels of improvisation, the current prototype is working with I₃ - the players are adjusting the structure in a manner sanctioned by the composer. This is very much a starting point - we intend to work towards I₇ and I₈ to allow use of the system as a truly interactive tool.

As more of the architecture is filled in, we gain the capability to experiment with more diverse situations. We envisage use of the system as a compositional tool, to create new works with a complexity and interdependence which would not be possible with human performers. We also intend the creation of interactive installations, which allow non-musical participants to become engaged in the creation of dynamic musical works.

REFERENCES

- [1] Álvaro Barbosa, 'Displaced soundscapes: a survey of networked systems for music and sonic art creation', *Leonardo Music Journal*, **13**, 53–9, (2003).
- [2] Katie Atkinson, Trevor Bench-Capon, and Peter Mcburney, 'A dialogue game protocol for multi-agent argument over proposals for action', *Autonomous Agents and Multi-Agent Systems*, **11**(2), 153–171, (2005).
- [3] Bruce Ellis Benson, *The Improvisation of Musical Dialogue*, Cambridge University Press, 2003.
- [4] Tim Blackwell, 'Swarm music: improvised music with multi-swarms', in *Proceedings of the 2003 AISB symposium on AI and Creativity in Arts and Science*, pp. 41–9, (2003).
- [5] Tim Blackwell and Michael Young, 'Self-organised music', *Organised Sound*, **9**(2), (2004).
- [6] Simon Dixon, 'A lightweight multi-agent musical beat tracking system', in *Pacific Rim International Conference on Artificial Intelligence*, pp. 778–788, (2000).
- [7] FIPA communicative act library specification. Foundation for Intelligent Physical Agents, www.fipa.org/specs/fipa00037, 2002.
- [8] FIPA Specification. <http://www.fipa.org/repository/bysubject.html>.
- [9] Andreas Herzig and Dominique Longin, 'A logic of intention with co-operation principles and with assertive speech acts as communication primitives', in *Proc. 1st Int. Joint Conf. on Autonomous Agent and Multi-Agent System (AAMAS 2002)*, pp. 920–927, Bologna, (15-19 juillet 2002). ACM Press.
- [10] Keiji Hirata and Tatsuya Aoyagi, 'Computational music representation base on the generative theory of tonal music and the deductive object-oriented database', *Computer Music Journal*, **27**(3), 73–89, (2003).
- [11] Fred Lerdahl and Ray Jackendoff, *A Generative Theory of Tonal Music*, MIT Press, 1983.
- [12] Evenadro Manara Miletto, Marcelo Soares Pimenta, Rosa Maria Vicari, and Luciano Vargas Flores, 'Codes: a web-based environment for cooperative music prototyping', *Organised Sound*, **10**(3), (2005).
- [13] D. S. Murray-Rust and A. Smaill, 'Musical acts and musical agents', in *Proceedings of the 5th Open Workshop of MUSICNETWORK: Integration of Music in Multimedia Applications (to Appear)*, (2005). preprint at http://homepages.inf.ed.ac.uk/s0239182/Murray-Rust_MusicalActs.pdf.
- [14] David S. Murray-Rust, 'Virtualatin: Towards a musical multi agent system', in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (to appear)*. IEEE CS Press, (2005).
- [15] Francois Pachet, 'Beyond the cybernetic jam fantasy: The continuator', *IEEE Computers Graphics and Applications*, (January/February 2004). special issue on Emerging Technologies.
- [16] Leo Kazuhiko Ueda and Fabio Kon, 'Andante: A mobile musical agents infrastructure', in *Proceedings of the IX Brazilian Symposium on Computer Music*, pp. 87–94, Campinas, Brazil, (2003).
- [17] Christopher D. Walton, 'Multi-agent dialogue protocols', in *AI&M 2004, Eighth International Symposium on Artificial Intelligence and Mathematics*, (2004).
- [18] Gil Weinberg, 'The aesthetics, history and future challenges of interconnected music networks', in *Proceedings of the International Computer Music Association Conference*, pp. 349–56, Göteborg, Sweden, (2002).
- [19] Ian Whalley, 'Piwecs: enhancing human/machine agency in an interactive composition system', *Organised Sound*, **9**(2), (2004).
- [20] M. Wooldridge, *Reasoning about Rational Agents*, MIT, Cambridge, MA, 2002.
- [21] Rodolfo Daniel Wulfhorst, Lauro Nakayama, and Rosa Maria Vicari, 'A multiagent approach for musical interactive systems', in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 584–591. ACM Press, (2003).

Bayesian modelling of colour's usage impact to web credibility

Eleftherios Papachristos¹ and Nikolaos Tselios² and Nikolaos Avouris³

Abstract. Colour plays an important role in web site design. The selection of effective chromatic combinations and the relation of colour to the perceived aesthetic and emotional value of a web site is the focus of this paper. The subject of the reported research has been to define a model through which to be able to associate colour combinations with specific desirable emotional and aesthetic values. The presented approach involves application of machine learning techniques on a rich data set collected during a number of empirical studies. The data obtained were used to train a Bayesian Belief Network which associated a simple chromatic model to perceived aesthetic and emotional dimensions. A set of tools that have been build in the process to support the methodological framework and ensure its reusability with minimal effort, are also described.

1. INTRODUCTION

The web has increased in complexity over the years and has gradually transformed from an online publication medium to a platform for carrying out complex tasks, such as shopping, learning, communication, and collaboration. Due to this transformation, additional research is required to support design of effective web sites. Various issues influence the effectiveness of a web site, such as structure, content hierarchy, styles, text format and navigational support. One of the most important design factors that influence the effectiveness of a web site is colour. Research and application of traditional human computer interaction methods and practices led to a significant improvement of web design quality. Yet, there is an open discussion about the specific elements that should be incorporated in a "well designed" web site. This question has particular significance during the development of web applications, considering the complexity of such a task [1].

Recent work [2], [3], has shown that the design space of a web site is affected by both engineering and aesthetic issues. Other work, suggests that the visual aesthetics of computer interfaces are a strong determinant of users' satisfaction and pleasure [4]. An effective web site design has to communicate well not only the content and the overall information architecture, but also wider values such as sense of professionalism, skilfulness and credibility. Furthermore, often the web site values are influenced by cultural beliefs, traditions, as well as goals and usage expectations [6]. The aforementioned criteria determine the *Perceived Value* of the site. Recent web based studies [5], showed a clear link between effective page design and credibility. Websites which seemed to have a more suitable visual appearance for the scope of the site were rated by the subjects as more credible. This reinforces the theory, established by social psychology research, that physically attractive sources have been perceived to be credible [7].

However, in general, there are not well established guidelines that affect the emotional aspects of interaction which, in turn,

influence a web site's perceived credibility. As a result, various interaction values are implemented via ad hoc procedures and trial and error implementations. So, there is a need for a systematic development of models that can guide the design practice in this area. Most modelling approaches supporting human interaction with web sites have focused primarily on the cognitive aspects of interaction [8],[9]. However, the importance of aesthetics on websites is well accepted and models of their impact on credibility have been proposed [10]. In general, however, little is known about the influence of aesthetics and the role of the colour. The most prominent designer's tool to use colour in websites is that of guidelines. However, existing colour guidelines are often awkward, easy to misinterpret and of limited value. According to Schwier and Misanchuk [11], experiential advice tends to be non-representative, contradictory or even obsolete. This is due to the outdated equipment used in the experiments, the fact that some guidelines may be only applicable on display media other than CRTs, and the tasks are so different and unique from common situations, that their generalization may be impossible. Even assuming that relevant guidelines exist, they are of little value to novice designers with little experience with colour theory and art. Many issues remain untackled by existing guidelines, such as the relationship between colours, the optimal number of colours for each case etc. Following this, it is desirable to obtain a methodology, which is capable of proposing specific directives about colour usage in any web design problem.

In this paper, we use Artificial Intelligence techniques to define a model to aid us select the most appropriate colour combination for a web site of a given purpose. The goal is to identify methodologically the colour combinations which communicate effectively desired emotional values. For example, a news web site should effectively communicate values such as consistency, reliability, objectivity, etc. Due to the fact that a direct evaluation of each colour is heavily subjective, the methodology tackles this problem by indirectly evaluating the credibility influence of selected colour combinations, and then breaking down the underlying influence of each factor using a Bayesian Belief Network. The developed Emotional Colour Model has been used in a tool to suggest appropriate chromatic schemes according to the desired perceived web site values.

2. EMOTIONAL COLOUR MODEL DEFINITION

The proposed model is partially inspired from a similar research by Guerin et al. [12]. This experiment examined colour usage, in the frame of interior space design. The subjects of the experiment had to evaluate computer generated interior environments against a list of 21 words, selected for this purpose on a scale from 1 to 5 (1 when a characteristic described by a specific word was rarely present, 5 when its presence was very strong). A variation of this experiment has been used to recognize differences in colour

¹ University of Patras, Greece, email: epap@ee.upatras.gr

² University of Patras, Greece, email: nitse@ee.upatras.gr

³ University of Patras, Greece, email: avouris@ee.upatras.gr

preferences between the two sexes [13]. The methodology proposed in this study is built upon the same idea, but is applied on web pages instead of interior environments. Our ultimate research aim is to determine relations between colour characteristics and descriptors of the web site's perceived value. The benefit of such an approach is the elicitation of a formal methodological process to select the appropriate colour combination. Furthermore, the process could be applied reversely: For a web site of a given scope we can elicit the appropriate colour characteristics.

A Colour Model has been defined first, combining physical, aesthetic and artistic dimensions. This includes aspects such as dominant and secondary colour, number of colours, dominant colour and secondary colour attributes such as degree of saturation and brightness (see Table 1). This model contains attributes related to the physical properties of the primary and secondary colour (e.g. colour hue, degree of saturation and brightness), as well as psychological properties (e.g. perceived warmth of the primary colour, colour scheme). Any web page colour scheme can be described according to this. The model cannot be conceived as too restrictive or exhaustively prescriptive, in terms of explicitly describing the very exact nature of each colour, which a web site's layout contains. This is compatible with our scope: The purpose of the proposed methodology is to guide the designer selecting appropriate colour schemes, not to suggest the exact attributes of each colour contained in the layout to be designed.

Table 1. Colour Model attributes with typical values

Attribute	Description	Values
Dom_colour	Hue of the dominant colour	Blue, Red, Grey, Green, Purple
Sec_colour	Hue of the secondary colour	Blue, Red, Grey, Green, Purple,
Dom_atr_dark	Brightness of the dominant colour	Dark, neutral, bright
Sec_atr_dark	Brightness of the secondary colour	Dark, neutral, bright
Dom_at_muted	Saturation level of the primary colour	Low, Neutral, High saturation
Sec_at_muted:	Saturation of the secondary colour	Low, Neutral, High saturation
Dom_atr_warm	Dominant colour's warmth	cold, neutral, warm
Sec_atr_warm	Secondary colour's warmth	cold, neutral, warm
Contrast_Values	Shows if contrast between brightness levels of main colours are high or low	high, neutral, low
Contrast_colours	Shows if contrast between hues of the main colours are high or low	high, neutral, low
Colour_sheme	Shows the type of chromatic combination used	Analogous, complementary, split-complementary, monochromatic
Numb_col	Total number of colours	(integer)

The Model is correlated to a number of descriptors, describing the perceived value of a web site. According to our methodology, typical users of the web site are requested to provide their subjective feeling about different colour schemes applied on representative web page layouts. A questionnaire has been designed for this purpose. By simply asking the participants for their preferences in colours, their answers would have been highly subjective, thus leading to non meaningful or contradictory conclusions. For example, in the case of a sports news web page, a person's preferences is heavily biased by social and geographic

influences (colour of local team, national team etc.). Therefore, we constructed an indirect approach to gather the perceived value of the site.

The Perceived web site value dimensions are based on a model used by Guerin, et al. [12], to describe the characteristics of interior environments. To reduce the ambiguity due to similarity between a number of the terms, the 12 most distinctive were chosen: *Pleasant, Formal, Fresh, Modern, Friendly, Aggressive, Professional, Attractive, Calming, Dynamic, Reliable and Sophisticated*. These values are compatible with the thirteen emotional dimensions that have been used by Kim et al. [14] to describe the emotional value of web site design, which were derived through a systematic process. The only dimensions missing from the model used in this study are those related to the web site structure and form (e.g. concise, simple) that are not related to the colour of a web page.

The product of the method described above is the *Emotional Colour Model*, a knowledge body that can be interrogated, in order to extract useful recommendations on colour usage. We do not expect generalized conclusions from this Model, at least unless a very large number of representative users are engaged. However, specialized strategies for colour usage in certain layouts may be obtained. For every new web site of a given scope and with recognized typical users in terms of characteristics like age, sex, educative level and familiarity with the internet, the methodology attempts to propose the most suitable colours to be used.

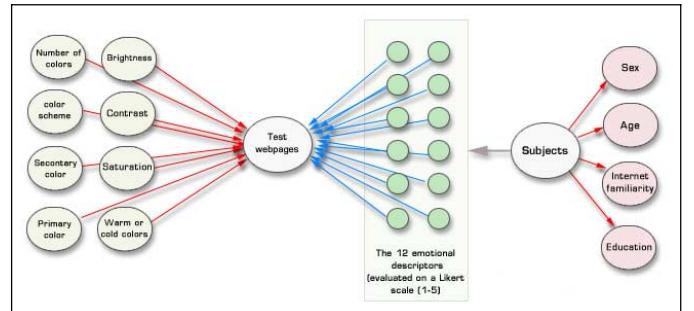


Figure 1. Overview of the proposed methodology

The data collected train a bayesian belief network. The variables of the network are not the layouts themselves, but their colour combination structural components. Thus, we can infer against most probable colour selections given specific credibility attributes which constitute the desired credibility for the designing web site to be designed. Reversely, given some colour scheme selections, we can predict the perceived credibility for given users.

Bayesian Belief Networks (BBN) are a significant knowledge representation and reasoning tool, under conditions of uncertainty. Given a set of variables $D = \langle X_1, X_2 \dots X_N \rangle$, where each variable X_i could take values from a set $Val(X_i)$, a BBN describes the probability distribution over this set of variables. We use capital letters as X, Y to denote variables and lower case letters as x, y to denote values taken by these variables. Formally, a BBN is an annotated directed acyclic graph (DAG) that encodes a joint probability distribution. We denote a network B as a pair $B = \langle G, \Theta \rangle$, (Pearl, 1988) where G is a DAG whose nodes symbolize the variables of D , and Θ refers to the set of parameters that quantifies the network. G embeds the following conditional independence assumption:

Each variable X_i is independent of its non-descendants given its parents.

Θ includes information about the probability distribution of a value x_i of a variable X_i , given the values of its immediate predecessors. The unique joint probability distribution over $\langle X_1, X_2 \dots X_N \rangle$ that a network B describes can be computed using (1):

$$P_B(X_1 \dots X_N) = \prod_{i=1}^N P(x_i | \text{parents}(X_i)) \quad (1)$$

In order to construct a BBN from the training data provided, two processes should be unified: learning the graphical structure and learning the parameters Θ for that structure. In order to seek out the optimal parameters for a given corpus of complete data, we directly use the empirical conditional frequencies, extracted from the data [15]. The probability that the model gives to the data can be extracted using the following formula proposed by Glymour and Cooper [16]. Then, we follow greedy search with one modification: instead of comparing all candidate networks, we consider investigating the set that resembles the current best model most. As we shall discuss in the conclusions section, BBN are a significant tool for knowledge representation, visualizing the relationships between features and subsets of them. This fact has a significant result on identifying which features actually affect the class variable, thus reducing training data size without any significant impact in the performance.

The proposed model building process is supported by a tool aimed at facilitating data collection. The collected data are stored first in a relational database, together with the attributes of the Colour Model for each evaluated layout. A second tool realized, to allow the designer choosing which perceived value descriptor has to be present and at what degree on the web site to be designed. According to the designer's preferences, the system proposes a list of colour characteristics, by using the BBN obtained. Further explanations on the results, such as information for colour theory elements, are also provided. The outlined methodology and use of the derived model and tool is further discussed in a validation case study in the next section.

3. CASE STUDY

Two empirical studies were conducted, to identify the quantitative relations between emotional dimensions and colour usage in a specific layout. In the first study, we tested the methodology with 46 participants in a highly controlled experiment. Our goal was twofold. Firstly, to examine the soundness of the proposed methodology. Secondly, to proof the validity of the used emotional descriptors. It was also a good opportunity to evaluate the tools developed to support the proposed methodology, as well as to improve them. The conclusions of this preliminary experiment found to be very encouraging. The tool developed to make use of the data collected could be a very helpful addition in the web designer's toolbox [17]. However, to claim direct facilitation of the designer's task to pick the most appropriate colours, evaluation of a wider range of layouts and colour schemes by a larger sample of subjects should take place. That is, because the data are hardly sufficient to derive results of general value. In the second study, we conducted another experiment in which the objective was to overcome the weaknesses unveiled in the first one. The first problem was to recruit a respectable amount of participants for the goals of this study. To tackle this problem, we converted the tool with which the participants were called to identify the quantitative relations between the emotional descriptors and colour characteristics to operate online (Figure 2). By doing so, we were

able to recruit more participants to carry out the evaluation but had also to deal with problems derived by a less controlled experiment. The tool developed for the experiment is a web application, which presents the layouts and the emotional dimensions on a Likert scale in the same window of a common web browser. The participants had to evaluate 10 out of 30 test layouts, in order to complete the experiment which is a sufficient number taking into consideration the user's weariness. Consequently, the application had to generate the layouts in a random order. Also the order of the descriptors appearance had to change from user to user, in order to avoid position related bias. Furthermore, the application has to provide sufficient documentation and techniques to avoid errors during the evaluation. Finally, it had to adapt to the users screen resolution and depict the appropriate layout.



Figure 2. Tool to acquire users' subjective evaluations

In order to represent in our study a larger area of colours, colour schemes and their derivatives, we had to increase the number of test layouts considerably. We considered that 30 different layouts comprised of six dominant colours with five variations in brightness, saturation, colour scheme etc. managed to balance a representative sample. By taking user weariness in consideration we decided that each participant should evaluate no more than 10 test layouts.

Table 2. Most friendly dominant colour

	First experiment	Second Experiment
1	Green	43.79 %
2	Blue	35.77 %
3	Grey	17.23 %
4	Purple	1.61 %
5	Red	1.60 %
6		Orange

Overall, 214 users (64 male, 150 female) participated in the experiment, evaluating a total of 10 randomly selected layouts from the 30 available, which totals an average of 70 evaluations for each layout. The median age was 18 years old. Most of the participants were undergraduate students from the department of Educational Sciences and Early Childhood Education Department as well as from the department of Electrical and Computer Engineering at the University of Patras. The majority of the participants had average (65,45%) or frequent experience (27,44%) with the Web.

Table 3. Most friendly brightness of the dominant colour

	First experiment	Second Experiment
1	Dark	4.31 %
2	Neutral	27.03 %
3	Bright	68.66 %

By comparing the data of the two experiments some findings seemed to be aligned. For example, the findings found to be similar both for the most “friendly” dominant colour, as shown in Table 2, and the attribute brightness of the dominant colour as shown in Table 3.

The network obtained showed no significant differentiation in preferences between the two genders in both experiments. In the first one, no connection was found between the node user’s gender and the other network, which lead us to believe that either our sample was inadequate or there is no difference in preference. In the second experiment the gender node found its way into the network but as shown in Table 4 there is only a slight difference between male and female preferences. This is mainly due to the unilateral sample of the participants.

Table 4. Differences between genders in dominant colour preferences

Men		Women	
Green	17.38 %	Green	17.43 %
Yellow	17.19 %	Yellow	17.30 %
Purple	16.59 %	Blue	16.50 %
Blue	16.36 %	Orange	16.32 %
Orange	16.27 %	Red	16.27 %
Red	16.21 %	Purple	16.18 %

By examining the structure of the tree, one may notice that the perceived values, the colour attributes and the demographical data were clustered into three large groups. The group of the perceived values has been placed at the bottom and the colour attributes at the top left of the tree of Figure 4. The only connection between the two sections of the model is made through the nodes “modern” and “saturation level of secondary colour”. According to BBN’s theory, if they’re no direct connections between two nodes then there is no direct relationship between them. However, a relationship may exist if they are connected indirectly, via other nodes, but the former case is an indication of a stronger relationship. Therefore, conclusions can be carried out, based on the position of every node and their “distance” from the Perceived Value descriptor nodes.

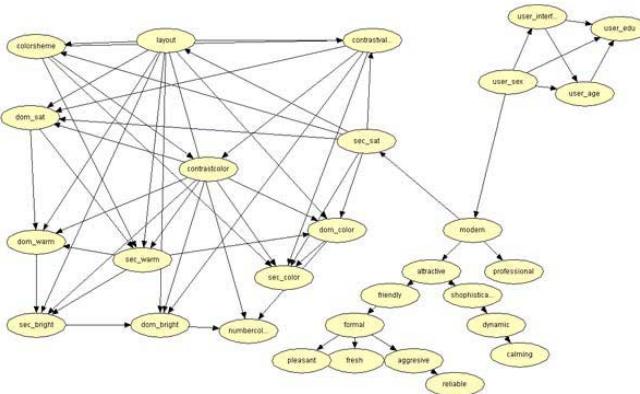


Figure 3. BBN obtained from user’s evaluations.

So, according to the tree structure the *saturation* of the secondary colour has a stronger effect on the site’s Perceived Value than its *hue*. Further analysis of the model structure, led to a categorization of influence of colour factors into 3 levels of importance: The ranking of the importance of the colour model aspects that influence the values perceived by the user seems to be

the following:

- First, the *saturation* of the secondary colour.
- Secondly, the *Hue* of the dominant and secondary colour, *saturation of the dominant colour*, the *colour scheme* and the *contrast levels of brightness*
- Finally, the saturation of the dominant colour, brightness of the dominant colour and the type of the dominant and secondary colour (warm or cold).

By comparing the trees generated by the two experiments we found slight changes in the nodes positions and dependencies. In the first experiment, the top node in the perceived value section was the node *friendly* and for the colour attributes the node *brightness of dominant colour*. This is due to the different physical layouts used. These users were more sensitive to these values for this particular layouts. This ascertainment stress the need to include a variety of test layouts in order to have results of generalizable value

Table 5. A web site’s colour model with respect to the emotional dimension ‘professional’

Attribute	“Professional” web page	Less “Professional”
Dominant colour	Green	Purple
Brightness of dominant colour	Bright	Dark
Saturation of dominant colour	High	Neutral
Secondary colour	Blue	Yellow
Brightness of secondary colour	Neutral	Dark
Saturation of secondary colour	Saturated	Neutral
Contrast of brightness between colours	Low	High
Contrast between Hues	High	Low
Number of colours	3	4
Colour Scheme	Complementary	Monochromatic

Besides observations based on the structure of the network, or performing sensitivity studies on various nodes, other interesting conclusions have been derived. It is possible to get information for every emotional dimension and for combinations of them from the model. For example, a colour scheme comprising no more than 3 colours seems to be the most appropriate to use, and the best chromatic combinations are complementary and analogical. In addition, it seems to be suitable to use low contrast levels between colours. Warm colours are preferable for dominant colours and cold for secondary. Furthermore, if we choose to adopt a monochromatic colour scheme it is preferable to have neutral contrast in brightness. Finally, regarding saturation, it was deduced that the best combination is high-saturated dominant colour with low saturated secondary colour. In order to demonstrate the use of the derived model, the proposed ideal colour factors for a web page required to communicate a sense of ‘professionalism’ is presented in Table 5. Additionally to this, the differentiation of those colour factors is presented, in case where the complete opposite emotional descriptor is required to be communicated.

The aforementioned analysis results do not comprise the only approach to take advantage of the knowledge embedded in the network. A great analysis tool is the observation of the quantitative dependencies between the nodes of the network. For example, by choosing a combination of emotional dimensions if we desire the presence of the descriptor “attractive” to a high extent, and the presence of the descriptor “professional” to a medium level, the network suggests ‘purple’ to be selected as a dominant colour with a probability of 17,43%. Setting dominant colour’s attribute value

to ‘purple’ the network led us to recommendations related to the other attributes. It suggested using bright warm and highly saturated purple, combined with a bright low saturated blue for secondary colour. It also posited usage of a complementary colour scheme but with no more than 3 colours of low contrast. Therefore, the solution space for the designer is significantly reduced to a certain amount of layouts complying with the requirements set in the described example.

4. CONCLUSIONS

In this paper, we proposed an innovative approach to define a model of a web site’s colour characteristics associated to desired perceived values. The proposed methodology involves a machine learning approach to create a belief network correlating the colour model of a web site with the values that are attributed to it by its users. We argue that the selection of appropriate colour schemes can lead to effective communication of the desired values for the web site users, thus enhancing the perceived credibility of the web site to be designed. In addition, we consider the proposed model-based approach to have significant advantages compared to the most common approaches, i.e. applying guidelines for colour usage.

The objective of the framework is to extract conclusions related to the specific colour selections and their emotional impact. To evaluate the proposed method we conducted two experiments: one preliminary with a small number of participants and a second one with a considerably large amount of subjects. However, the results reported in this paper, cannot be claimed to be of general value, since neither the number of subjects, nor the number of the test web pages was sufficient to justify general conclusions from the conducted studies. Despite that, it should be stated that many of the findings reported here, confirm earlier empirical rules related to colour’s usage and current tendencies in web design, which is an encouraging indication of the validity of the proposed methodological approach.

As for future research, we intend to extend the nature of the forthcoming experiments by altering the physical layout of the used template by differentiating parameters such as different layouts, font sizes, white space used etc. in order to examine the effect of those parameters to the results obtained. Additionally, we intend to use the proposed methodology on actual web sites. Subsequently, to perform other experiments in which to make use of the developed models for the design of specific kinds of web sites, like e-learning applications. With regard to the colour model used, several attributes seem to have more practical usefulness than others, while using the methodology reversely, since the proposed colour model might be considered insufficient by many practitioners involved in web design.

Finally, as an extension to the reported study, one may view the issue of appropriate chromatic combination to a broader context and examine its impact to the perceived information gain in particular environmental conditions in dynamic Internet ecology [18]. We argue that various models of typical web user behaviour, like information foraging [19], based on cognitive models combined with AI techniques, should be extended to take into account colour issues.

ACKNOWLEDGEMENTS

This research was funded by EPEAEK 2: ‘Pithagoras 2: Models of information foraging on the web’. Substantial appreciation should be paid to Dr. Manolis Maragoudakis for his valuable insights in Bayesian Belief Networks application on our problem.

REFERENCES

- [1] Avouris N., Tselios N., Fidas C., Papachristos E., Website evaluation: A usability-based perspective, in Manolopoulos Y. et al. (ed.) Advances in Informatics, LNCS No 2563, pp. 217-232, Springer Verlag, Berlin, 2003
- [2] Lavie, T. and Tractinsky, N. (2004). Assessing Dimensions of Perceived Visual Aesthetics of Web Sites, International Journal of Human-Computer Studies, 60(3):269-298.
- [3] Karvonen, K. (2000). The beauty of simplicity, Proceedings on the 2000 conference on Universal Usability, p.85-90, November 16-17, 2000, Arlington, Virginia, United States.
- [4] Tractinsky, N., Shoval-Katz A. and Ikar, D. (2000). What is Beautiful is Usable. Interacting with Computers, 13(2): 127-145.
- [5] Danielson, D.R. (2003). Transitional volatility in web navigation. Information Technology and Society, 1(3), Special Issue on Web Navigation, B. Shneiderman, J. Lazar, and M.Y. Ivory (Eds.), 131-158.
- [6] Krug, S. (2000). Don’t make me think: a common sense approach to web usability. Macmillan USA.
- [7] Benoy, J W. (1982). The credibility of physically attractive communicators: A review. Journal of advertising, 11(3), 15-24
- [8] Fuccella, J., (1997). Using user centered design methods to create and design usable web sites. Proceedings of Annual ACM Conference on Systems Documentation, pp. 99-101.
- [9] Wang, P., (2001). A survey of design guidelines for usability of web sites. Proceedings of the 2001 Human-Computer Interaction International Conference 1, 183-187.
- [10] Fogg, B.J., C. Soohoo, D. R. Danielson, L. Marable, J. Stanford E. R. Tauber (2003). How do users evaluate the credibility of web site? A study with over 2500 participants, Proc. Conference on Designing for user experiences, pp. 1-15, San Francisco, California
- [11] Schwier, R.A. and Misanchuk, E.R. (1995). The art and science of colour in multimedia screen design Proceedings of Annual Conference of the Association for educational communications and technology, Anaheim, CA.
- [12] Guerin, D. A., Park, Y., & Yang, S. (1995). Development of an instrument to study the meaning of colour in interior environments. Journal of Interior Design, 20 (2), 31-41.
- [13] Khouw N., (2004). The Meaning of Colour for Gender. Internet source, available at <http://www.colournatters.com/khouw.html>.
- [14] Kim J., Lee J., Choi D., (2003), Designing emotionally evocative homepages: an empirical study of the quantitative relations between design factors and emotional dimensions, Int. J. Human-Computer Studies, 59, pp. 899–940.
- [15] Cooper J., Herskovits E. (1992). A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9, pp.309-347.
- [16] Glymour C., Cooper G. (eds.). (1999). Computation, Causation & Discovery. AAAI Press/The MIT Press, Menlo Park.
- [17] Papachristos E., Tselios N. Avouris N., E. Papachristos, N. Tselios, N. Avouris, Inferring relations between colour and emotional dimensions of a web site using Bayesian Networks, Proc. Interact 2005, Rome, September 2005
- [18] Furnas, G. W, (1997). Effective view navigation. In Proceedings of the Human Factors in Computing Systems, CHI '97 (pp. 367-374). Atlanta, GA: ACM.
- [19] Pirolli, P. and Card, S. K. (1999). Information Foraging. Psychological Review 106(4): 643-675.

Evaluating Perception of Interaction Initiation in Virtual Environments using Humanoid Agents

Christopher Peters¹

Abstract. We evaluated user perception of certain attention behaviours, eye head and body orientation in particular, of humanoid agents in a virtual environment for the purposes of interaction initiation. Users were shown a number of scenarios involving agents making gaze, gesture and locomotion behaviours and were asked to report their impressions of how attentive the agents were and their perceived likely-hood to engage in interaction. These evaluation studies are critical for furthering our understanding of the early phases of human-computer interaction where distance may be involved and there is uncertainty as to the intention of the other to interact. Amongst other results, we establish here that the human perception of interest, interaction seeking and openness from a humanoid agent is based, in part, on the time-varying behaviour of its eyes, head, body and locomotion directions. Application domains include social robotics and embodied conversational agents.

1 INTRODUCTION

In this paper, we describe an experiment to determine the extent to which agents in virtual environments give a human user the impression that they are attentive or interested in them, in particular, by orienting their gaze, body orientation and locomotion with respect to the user. We furthermore evaluate how, in the absence of other modalities of communication (e.g. verbal or facial expression), such behaviours may be interpreted by the human as a sign by the agent that it is open to or wishes to become involved in a more in-depth interaction with the user e.g. start a conversation with them.

The start of interaction is an important phase in human-computer interaction, *HCI*, and one that has thus far been largely neglected in the literature. Many ‘start of interaction’ scenarios in *HCI* consist of an inflexible stage occurring at short-distance, where either of the participants have already been identified as having the goal of interacting with each other, or in other cases, one participant has the task of fulfilling a number of inflexible constraints and overt cues in order to open with the other. For example, a human may have to stand in a predefined position in front of the computer, press a button or make an utterance in order to activate it. This approach avoids many complex issues but, unfortunately, also disregards the grace, naturalness and subtleness with which real human interactions tend to begin. Our work represents a first attempt to allow for a more natural start to the process of interaction opening. The design of such a system will no doubt involve a wide range of concepts in fields that are still developing, including computational modelling of visual attention and theory of mind.

In this paper, we evaluate some fundamental aspects of how humans *perceive* attention and related behaviours from humanoid

agents in both static and dynamic scenarios. Research concerning this domain is a veritable jigsaw of varying interrelated topics spanning a number of fields over a wide time-line, from sociology to evolutionary psychology to neuroscience; holistic, macro-level research that is of use in the construction of practical *HCI* systems is still lacking. In this respect, we provide a first practical evaluation towards the construction of such a system. Our particular application domain is to support the cognitive modelling and animation of certain autonomous agent behaviours, such as gaze (see for example [11] and [6]).

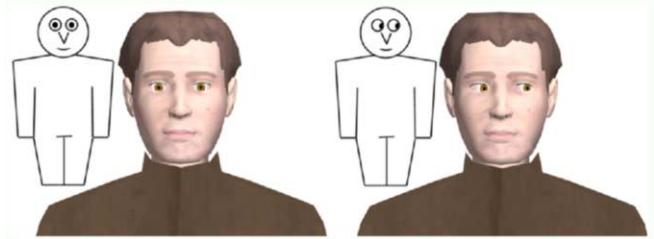


Figure 1. A close-up of the humanoid agent used in the evaluation studies. Beside each agent is an illustration of the eye, head and body directions.

2 RELATED WORK

A number of studies have established some fundamentals regarding the human perception of humanoid agents and avatars for up-close interaction. Colburn et al. looked at how eye-gaze animations affected the perception of humanoid agents and found that a natural gaze model elicited changes in a human viewers’ eye gaze patterns [2]. Garau et al. studied the impact of the graphical realism level of an avatar and how it affected the quality of communication [4]. They found that an informed-gaze avatar significantly outperformed a random gaze avatar and also that alignment of behavioural and visual realism was important for creating effective avatars.

Turk et al. manipulated the gaze of avatars in order to change user’s perception of attention in immersive virtual environments and found large gender differences in presenter awareness, persuasion and memory between different gaze conditions [13]. Vertegaal et al. [14] considered the significance of gaze and eye contact in the design of GAZE-2, a video conferencing system that ensures parallax-free transmission of eye-contact during multiparty mediated conversation. More recently, and of great interest to our studies, Sidner et al. [12] have studied rules of looking behaviour in order to allow robots to maintain engagement with humans in a collaborative environment. They found that users engaged in mutual gaze with the

¹ LINC Laboratory, University of Paris 8, email: c.peters@iut.univ-paris8.fr

robots, directed their gaze to them during turns in conversation and responded to changes in head and gaze direction by changing their own gaze or head direction.

3 BACKGROUND

Our work pays homage to the role of the eyes, gaze and attention in the start of interaction. There are numerous reasons as to why vision is important for the early detection and signalling of willingness to interact. For example, in the social sciences, Kendon [7] analysed human greetings, describing a sequence of opening cues for meeting interaction and noting interaction fundamentals, such as the requirements for participants to first sight each other and identify the other as someone they wish to greet. Goffman further noted [5] that we generally seek to avoid the social embarrassment of engaging in interaction with an unwilling participant: thus, even if we have identified the other as one we wish to interact with, if we are to limit the possibility of social embarrassment (suffered, for example, if the other were to completely ignore us), it is a wise strategy to first establish with some degree of confidence that they will reciprocate in the interaction before we become too explicit in our actions. Gaze negotiation may be one such method of establishing such confidence in a subtle manner, while at the same time having the benefit of orienting the senses towards the object of communication. In the field of evolutionary psychology, the special importance of the eyes and perception of visual attention has been noted in theory of mind research [1]. This work has been supported by neurophysiological evidence [9] and it is particularly in this area where we wish to evaluate social gaze research [3] using computational models [10] in a practical scenario.

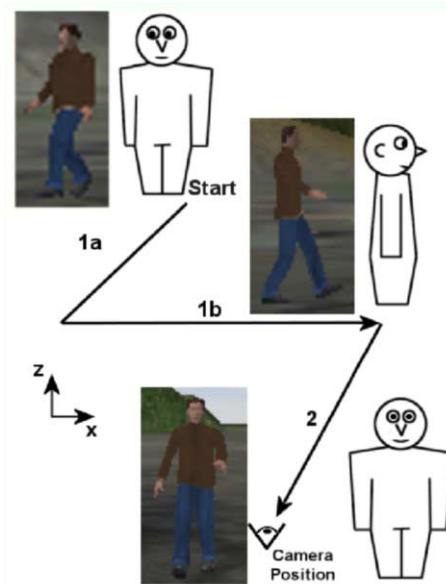


Figure 2. Path taken by the agent in the virtual environment. The path was divided into two main segments (1 and 2). During each DES trial, the agent followed a number of these segments, in order 1a, 1b, 2. While walking along segment 1a, the agent was at an oblique angle to the camera. In contrast, when walking along segment 2, the agent directly faced the camera.

4 EXPERIMENTAL DESIGN

A total of 21 participants engaged in a two-stage evaluation process. The first evaluation, which we will refer to as the static evaluation study, or *SES*, consisted of participants being shown a sequence of 25 static images featuring a humanoid agent standing in an upright posture. In each trial, body segments of the humanoid agent were oriented in varying positions with respect to the viewer: in some images, only certain body segments were visible. After viewing the image, participants were asked to select, using a slider, the amount of attention that they felt the agent was paying to them (“Attention which is being paid to me”), from a range of *NONE* to *A LOT*. The default position of the slider was a center point of the scale. Participants were then required to click a button in order to proceed to the next trial image. The ordering of the trials in the sequence was randomised for each participant.

In the second evaluation, referred to here as the dynamic evaluation study, or *DES*, participants were shown a sequence of 10 animations featuring a humanoid agent moving around in a virtual environment and making a range of behaviours. The behaviours of the agent ranged from ignoring the user, to looking at them, walking towards them and waving. After each trial, the participants had to adjust a number of sliders indicating how they interpreted the actions of the agent in terms of e.g. the amount of interest it had shown towards them and whether they thought the agent had seen them (see Table 1 for full listing). Participants then clicked a button to proceed to the next trial. Prior to the *DES*, participants were shown a test animation demonstrating the capabilities of the agent and what they were about to witness. We distinguished between two cases of an agent that was perceived to ‘want to talk’ to the participant (Table 1: *T*) and one that was thought ‘would respond’ (Table 1: *R*) to a talk request if made by the participant. The first case is suggestive of an agent that was perceived to be proactive in seeking interaction, while the second was a measure of the openness of the agent to interaction, but in a more passive sense.

Table 1. Evaluation sliders to be set after each trial of the DES.

	Text (The agent ...)	Type / Scale
(L)	looked at me	Yes / No
(K)	knows that I am here	Yes / No
(I)	is interested in me	Not at all ↔ Completely
(T)	would like to talk to me	Not at all ↔ Completely
(R)	respond to my request to talk	Not at all ↔ Completely
(F)	intention towards me is	Bad / Harm ↔ Good / Help

4.1 Goals and Hypotheses

This experiment had a number purposes. The most fundamental of these was to verify that human viewers could in fact infer attention and interest from an artificial humanoid character when viewed in a virtual environment. When a humanoid character looks at one, does one have the sense that they are the object of some form of attention? There were also more precise goals for each study. First of all, from the *SES*, to test the notion that the amount of attention that users perceive from an agent in a virtual environment is related to the manner in which three main body segments (the eyes, the head and the body) are oriented with respect to the viewer. This is inspired by research from ethology [3] suggesting that a similar situation may exist in nature, whereby the eyes, head, body and locomotion direction all

influence the perception of the amount of attention that one has in the self. Our initial hypothesis for SES was that the attention perceived by the user would increase as the orientation of agent subparts were directed more towards to user, with the eyes as the biggest indicator of attention, followed by the head and finally the body. Secondly, from the DES, we aimed to test human perception of the intention of an agent to start interaction based only on their direction of gaze and locomotion. Our hypothesis was that increased directed gaze, locomotion and gesture behaviour would result in an increased perception of an agent that was interested in the viewer and open to or seeking interaction.

4.2 Population

All participants were French computer science undergraduate students, between the ages of 19 and 25 and all therefore had a technical background. One sequence of results was disqualified from the SES due to all entries having a zero value and three were disqualified from the DES due to participants following the evaluation guidelines incorrectly. Of the 21 participants, 3 were female.

4.3 Apparatus

The study was conducted in two separate labs. Software was installed on six standard desktop PC's and tests were run in parallel, with six participants involved in the study at any one time independently of each other. All machines were similar in capability, with 3 GHz Intel CPUs running with 1GB of RAM and using Windows 2000 OS with a video resolution of 1280 x 1024 in 32 bit colour. In terms of viewing equipment, four of the machines had identical 19" *Belinea 10 19 20* flat-screen monitors with viewing areas of approximately 15" x 12", the other two monitors had viewing areas of approximately 14.5" x 11" and 13" x 10" respectively. Participants were seated between 40cm and 70cm from their screens. For the SES, the participants had as long as they needed to view the images and select a response. For the DES, participants only viewed an animation once before being presented with the questionnaire screen.

4.4 Software

The Torque engine (<http://www.garagegames.com>) was used to provide the three dimensional virtual environment and other graphics capabilities and to collect evaluation information from participants. The demonstration and collection process was fully automated, although a regulator was on hand to observe correct following of experimental protocol and to answer questions. Participants were initially presented with an in-engine screen providing instructions in French, their native language. From this screen, the participants were then guided through the entire evaluation process via menus, which also prompted for results. For the DES in the virtual environment, since the visibility of the agent's behaviours might be difficult for the viewer when the agent was far away due to the nature of the display equipment, we placed a magnified view of the agent in the top right corner of the display.

4.5 Virtual Agent and Environment

For the DES, participants' avatars were placed in a predetermined position in the virtual environment. They could not alter this position or move around in the environment, but could rotate their viewpoint through the use of the mouse. We used a single male agent model

in all of the cases (see Figure 1). In each trial, the agent started in a set start position (see Figure 2) and walked along a predefined path. The first sub-segment of path (1a) was directed so that the agent would move closer to the viewer without walking directly towards them. The second sub-segment (1b) was positioned so that the agent would walk perpendicular to the viewer. The second segment of path (2) was positioned so that the agent would have a locomotion direction directly towards the viewer. During trials, the agent would either walk along the first segment of path (1), or else both segments of path (1 and 2). A variety of gaze, locomotion and gesture behaviours were made during the trials: thus, trials differed in gaze (gaze at / not gaze at), gesture (wave / no wave) and locomotion direction (oblique / perpendicular / towards) with respect to the viewer.

5 RESULTS

Separate results were obtained for the two evaluation studies, SES and DES, as follows.

5.1 Static Evaluation Study

The averaged results of the static evaluation study for all 21 participants are summarised in descending order according to the amount of attention as perceived from static images of the humanoid agent (see Table 2). Values have a range of 0.0 to 1.0. Participants were asked to adjust a slider that indicated the amount of attention they thought the agent was paying to them in each instance: the slider ranged from *NONE* to *A LOT*. *NONE* has been mapped onto 0 and *A LOT* onto the value of 1.

Table 2. Results for the Static Evaluation Study (SES) in descending order according to the mean amount of attention that participants reported as perceived from the static images of the humanoid agent. The agent was segmented into three main parts: eyes, head and body. Direction was encoded as facing forwards towards the viewer (F), midway (M) and to the side (S).

	Eyes	Head	Body	Av. Rating	Std. Dev.
(a)	F	M	S	0.758	0.186
(b)	F	M	F	0.744	0.169
(c)	F	F	S	0.727	0.293
(d)	F	M	M	0.710	0.163
(e)	F	F	M	0.589	0.233
(f)	F	F	F	0.507	0.322
(g)	M	M	M	0.466	0.302
(h)	M	M	F	0.372	0.322
(i)	S	F	F	0.277	0.269
(j)	S	S	F	0.221	0.240
(k)	S	S	S	0.192	0.278

5.2 Dynamic Evaluation Study

The results for seven of the DES tests are shown in Figure 3. Each test shows, on the left, the path (corresponding to Figure 2) taken by the agent as well as icons indicating the timing of behaviours made along the path. On the right is a graph of the mean values over the 21 participants for each of the queries (Table 1: *I*, *T*, *R*, *F*). Values have a range of 0.0 to 1.0, where 0 is mapped onto *NOT AT ALL* for *I*, *T*, *R* and *BAD / HARM* for *F*. A summary of the results, in descending order for each category, is provided in Table 3.

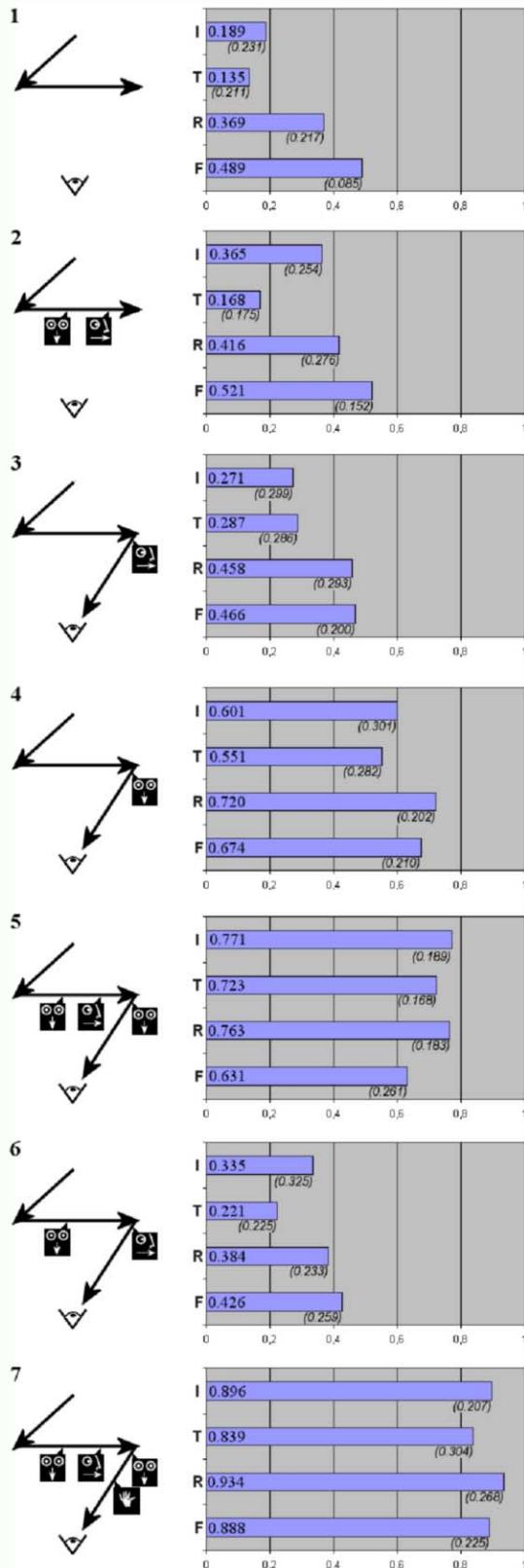


Figure 3. DES results for 7 cases. For each case, a top-down view of the path is presented on the left side, tagged with behaviour icons corresponding to look at viewer, look away from camera and wave gesture respectively and marked at the time at which they occurred. On the right are mean values for perceived *interest* (I), *want to talk* (T), *would respond* (R) and *intention* (F) accompanied, in brackets, by the standard deviation for each data set.

Table 3. Averaged perceived values from DES in descending order as reported by participants. Columns correspond to rows (I), (T) and (R) in Table 1. Cases and values correspond to those illustrated in Figure 3.

Interest (I)		Want Talk (T)		Would Respond (R)	
Case	Value	Case	Value	Case	Value
7	0.89	7	0.83	7	0.93
5	0.77	5	0.72	5	0.76
4	0.60	4	0.55	4	0.72
2	0.36	3	0.28	3	0.45
6	0.33	6	0.22	2	0.41
3	0.27	2	0.16	6	0.38
1	0.18	1	0.13	1	0.36

6 DISCUSSION

At a fundamental level, it is evident from our studies that virtual humanoids have the ability to give users the impression of being paid attention to through gaze, body orientation, gesture and locomotion behaviours. In addition, participants tended to report higher perceptions of interest, interaction seeking and openness overall from the agent when these behaviours were directed towards the user: In the SES, the case rated lowest in terms of the amount of attention perceived to be paid by the agent is that where all body segments are oriented away from the camera. Similarly, in the DES, the case rated lowest in all categories related to interest and interaction was case 1 (Figure 3; see also Table 3). It was in this case that the agent did not look at, walk towards or gesture towards the camera at all.

6.1 Static Evaluation Study

In general, from Table 2, it can be seen that participants' perception of attention from the humanoid agent was highly correlated with the agents eye direction: that is, when the agent was looking forward into the camera (giving the impression of looking at the user), participants rated it highly in terms of paying attention. Head direction correlated less strongly than eye direction, whereas the correlation between perceived attention and body direction was very low. Generally, these results adhere to the hierarchy put forward in [3] and used in [10], which is of the form eyes > head > body.

The most surprising result of this study was the relatively low ranking of 0.507 for case (f) (Table 2), the case where all of the body segments of the agent faced the user (Eyes: front, Head: front, Body: front). One would have expected this to be one of the highest ranked situations. A closer inspection of the results for this case revealed that participants were divided into two distinct categories: of the 21 participants, 9 provided a rating between 0.0 and 0.336 while 8 rated it between 0.654 and 1.0. Considering that, overall, higher ratings were awarded when the eyes were forward and the head/body were not, it is possible that these cases made an 'attentive signal' more apparent due to the contrast of the head/body directions with that of the eye direction. Thus, the lower rating group in case (f) may have perceived less of a signal from the agent e.g. interpreted it as a blank stare, due to lower contrast between segment directions and thus a less obvious signifier of attention in static images.

6.2 Dynamic Evaluation Study

Comparing cases 1 and 2 (Figure 3), where the only difference is a brief glance at the user, participants recorded an increased perceived level of interest from the agent (Interest: 0.189 → 0.365). In case

2, where the agent did not change locomotion direction after glancing at the camera, participants had the impression that the agent was somewhat interested in them (Interest: 0.365), would be responsive to conversation (Would Respond: 0.416), but not that the agent was actively trying to start an interaction (Want Talk: 0.16). Cases 3 and 4 provide a similar situation in terms of interest (0.271 → 0.601) and would respond to an interaction request (0.458 → 0.720). Unlike the previous situation, there was also an increase in the perception that the agent wants to talk (0.287 → 0.551). These results seem to suggest that, although perceived openness is closely linked to the perceived interest through gaze, the coupling between the perception of interest and perception that the agent wants to actively start an interaction requires more overt cues signalling this intention: in this case, a change of locomotion direction towards the camera. This is again supported in a comparison of cases 2 and 5: the agent glances at the user in both cases, but in case 5, it then changes locomotion direction towards them. This locomotion change results in a large increase in all reports (Interest: 0.365 → 0.771, Want Talk: 0.168 → 0.723, Would Respond: 0.416 → 0.763), particularly the impression that the agent actively wants to talk. Without the presence of any gesture behaviours, case 5 was rated highest. Ethologically too, this would seem to be one of the most natural behaviours preceding the start of an interaction: from the user's perspective, the agent walks perpendicular to them without initially being aware of them. It glances over, becomes aware of them, makes a decision to engage them and so, changes direction, walks towards them into interaction range and starts a conversation. The presence of a waving gesture seems to further concretise the perception of attention and reaffirm the intention of the other to engage in interaction (case 7).

In terms of the intention of the other, gaze and locomotion behaviour alone were able to influence participants' perception of the valence of the agent's intention towards them. Comparing cases 4 and 5, where the agent looks towards the participant as it approaches them, with cases 3 and 6, where the agent looks away, reveals a difference of 0.208 between cases 3 and 4 and 0.206 between cases 5 and 6. In each set, the only difference in behaviour was whether the agent looked towards the participant as it moved towards them. In both cases, when the agent moved towards the participant without looking at them, its intention was judged nearer the BAD / HARM end of the scale. In addition to looking towards the user while moving towards them, and despite the absence of facial expression in our studies, unsurprisingly, the agent was perceived as having the best intentions when a waving gesture was present (Figure 3, case 7). Despite the fact that participants reported the agent as not having looked at them, in many cases, they nonetheless reported that they thought the agent was aware of their presence. This may indicate that people attribute similar visual perception capabilities, such as field-of-view, to humanoid characters as to real humans (see case 3).

7 CONCLUSIONS AND FUTURE WORK

We have established a number of factors in this study which have not been the subject of previous research. First of all, virtual humanoids have the ability to give human users the impression that they are being paid attention to through their gaze, body orientation, gesture and locomotion behaviours. Secondly, the human perception of interest, interaction seeking and openness from the agent is based on the direction of eyes, head, body and locomotion of the agent and its dynamic behaviours. The studies here show that a perception of attention metric concerning different body segments, e.g. [10], needs to place a very high emphasis on the eyes of the agent. Beyond this,

more research is needed to establish the exact contributions of the head and body segments to the perception of attention. Our work could also be improved by using a better participant population: in particular, most of the participants in this experiment were male, and it is likely that there are important gender differences when interpreting the behaviour of others [8]. This work may also provide some illumination in areas of human-robot interaction initiation [12], although we are careful to stress the likely-hood of differences between the three cases of users perceiving from the real-world, perceiving a virtual environment using a flat-screen monitor, and perceiving while fully immersed in a virtual reality environment.

ACKNOWLEDGEMENTS

We would like to thank the referees for their insightful review comments and Catherine Pelachaud for her invaluable advice throughout this work. This work has been funded by the Network of Excellence Humaine (Human-Machine Interaction Network on Emotion) IST-2002-2.3.1.6 / Contract no. 507422 (<http://emotion-research.net/>).

REFERENCES

- [1] S. Baron-Cohen, 'How to build a baby that can read minds: cognitive mechanisms in mind reading', *Cahiers de Psychologie Cognitive*, **13**, 513–552, (1994).
- [2] A. Colburn, M. Cohen, and S. Drucker, 'The role of eye gaze in avatar mediated conversational interfaces', Technical report, MSR-TR-2000-81, Microsoft Research, (2000).
- [3] N.J. Emery, 'The eyes have it: the neuroethology, function and evolution of social gaze', *Neuroscience and Biobehavioural Reviews*, **24**(6), 581–604, (2000).
- [4] M. Garau, M. Slater, V. Vinayagamoorthy, A. Brogni, A. Steed, and M.A. Sasse, 'The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment', in *Proceedings of the conference on Human factors in computing systems*, pp. 529–536. ACM Press, (2003).
- [5] E. Goffman, *Behaviour in public places: notes on the social order of gatherings*, The Free Press, New York, 1963.
- [6] D. Heylen, I. van Es, A. Nijholt, and B. van Dijk, *Advances in Natural Multimodal Dialogue Systems*, volume 30 of *Kluwer Series on Text, Speech and Language Technology*, chapter 11, 245–262, Kluwer Academic Publishers, 2005.
- [7] A. Kendon, *Conducting interaction: patterns of behaviour in focused encounters*, Cambridge University Press, New York, 1990.
- [8] A. Kleinsmith, N. Bianchi-Berthouze, and L. Berthouze, 'An effect of gender in the interpretation of affective cues in avatars', in *Workshop on Gender and Interaction'06, In conjunction with the International Conference of Advanced Visual Interfaces*, Venice, Italy, (May 2006).
- [9] D.I. Perrett and N.J. Emery, 'Understanding the intentions of others from visual signals: neurophysiological evidence', *Current Psychology of Cognition*, **13**, 683–694, (1994).
- [10] C. Peters, 'Direction of attention perception for conversation initiation in virtual environments', in *International Working Conference on Intelligent Virtual Agents*, pp. 215–228, Kos, Greece, (September 2005).
- [11] I. Poggi, C. Pelachaud, and F. de Rosis, 'Eye communication in a conversational 3d synthetic agent', *AI Communications*, **13**(3), 169–182, (2000).
- [12] C.L. Sidner, C.D. Kidd, C. Lee, and N. Lesh, 'Where to look: a study of human-robot engagement', in *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pp. 78–84, New York, NY, USA, (2004). ACM Press.
- [13] M. Turk, J. Bailenson, A. Beall, J. Blascovich, and R. Guadagno, 'Multimodal transformed social interaction', in *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pp. 46–52, New York, NY, USA, (2004). ACM Press.
- [14] R. Vertegaal, I. Weevers, C. Sohn, and C. Cheung, 'Gaze-2: conveying eye contact in group video conferencing using eye-controlled camera direction', in *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 521–528, New York, NY, USA, (2003). ACM Press.

Cognitive situated agents learn to name actions

Julien Poudade and Lionel Landwerlin and Patrick Paroubek¹

Abstract. We present experiments in which a population of situated autonomous cognitive agents learns by means of “language games” a common lexicon of symbols that designates basic motions (translation, rotation) in a 2D world. We study the quality of the lexicon learnt when we give our agents a cognitive functionality akin to working memory, and we explain how they are able to agree on a set of common names for basic actions, if their world representation is built using sensorimotor contingencies informations.

1 Introduction

Up to now, all the attempts that have been made for giving a machine the ability to understand a language have consisted in providing it with the corresponding interpretation function. This means that if the machine is expected to interpret a message about its environment, it must then be given what is generally called “common sense” information. Since coding the world has revealed to be a hopeless task, except for a few extremely limited cases, we propose here the first fundaments of an approach where a machine could develop a language from sensorimotor categories it has discovered by itself while interacting with its environment, including machines like itself (unsupervised situated autonomous learning). Our initial aim is to identify the conditions required for the emergence of a language under this hypothesis. We decided to use variation of the naming game developed by L. Steels [14] as communication protocol to test our ideas. With these games, we studied how a population of autonomous agents develops a common lexicon, first to designate static objects then in a second set of experiments, to designate basic motions (rotation, translation) in a 2D world (Figure 1).

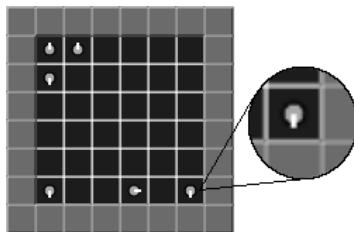


Figure 1. View of the simulation game board: circles are agents, (the light dash give their direction) darkened squares are free locations to which agents can move to, and light squares are obstacles which delimit the game area.

In addition to shedding some light on the fundamental principles required to validate our approach, we look in our experiments at the influence that a cognitive structure akin to “working memory” has on

the learning process. In this paper, we explain first how we think it is possible to perform meaning grounding through interaction with the environment, and describe the effects that such grounding has on the representations built by the agents. On this occasion, we make a short recall on affordances [5] and sensorimotor contingencies [10], two notions coming from precursory works which are essential to our enterprise. Then we describe the architecture of our agents and the different naming game protocols that we have used. The discussion of our experimental results for both the experiment on static object naming and basic action naming precedes the conclusion in which we recall the three results we have obtained so far.

2 Naming Games and Working Memory

Language game models [8] are assimilable to models of communication between agents of a population discoursing about their environment. Our interaction protocol, was inspired by the “naming game” [14]. In this game, agents communicate with each other by pointing objects to other agents and by enunciating words built from a common finite set of syllables. An environment is made of a set of objects and a population of agents². The lexicon is a set of pairs of word-object association, considered at the level of an agent or globally over the whole population. Each agent has an associative memory accessed by content, holding a representation of its lexicon, initially empty. This representation records the credit that the agent gives to each word-object association. As credit function we use the ratio of the number of times the association has successfully been used in a game over the total number of time it has been selected to take part in a game. The notion of working memory in humans has been introduced by [6] then redefined by [2] as being a multiple-component master-slave system. He showed throughout various experiments that the working memory influences everyday cognition activities, like reading apprenticeship, language understanding and vocabulary acquisition. We have completed the model of agent described above by adding a working memory (or short term memory) of fixed size. It contains the elements of the long term memory made salient by the current situation or the result of a cognitive computation.

In [12], we measured the effect that the working memory size has on the ambiguity factor of the lexicon built by our cognitive agents [4] in the course of a series of language games for learning to name objects. Even in an unambiguous closed world, i.e. a world where objects are uniquely distinguished from one another and the number of object is finite and enumerable, the event combinatorics of the language interactions generates ambiguous representations (homonymy, synonymy). It has been shown that in some cases up to 8 words can be used to reference the same object [12]. In these past experiment about the designation of static objects, it has been proved that the ambiguity factor is directly linked to the size of the working memory

¹ LIMSI-CNRS, Orsay, France, email: {poudade,landwerlin,pap}@limsi.fr

² Both sets are finite in our experiments.

of an agent. By increasing a little the size of the working memory, we drastically decrease the size of the lexicon, while the number of objects named remains constant. It is the number of word-object associations for the same object that diminishes.

Consequently, it has been shown that the size of working memory has a strong decreasing effect on the ambiguity factor of lexicon shared by the whole population of agents. Although the model of language game that was used was rather simplistic, the results obtained led us to think that the cognitive structures play an essential role in the acquisition of the symbols of a language. Now convinced of the role played by cognitive structures and of the necessity to define meanings through interaction with the environment, we can address the problem of perceptive categorization raised by [7]. To this end, we will now have a look at affordance, contingencies and semantic representations.

3 Affordances, Contingencies and Meaning

The notion of affordance or action grounding is initially borrowed from "the Gestalt" and can claim its origin in the ecologic psychology of [5]. The central idea behind the notion of affordance is that we perceive directly the functional value of objects, their practical meaning : what is appropriate to do, the risks and obstacles. Affordances are characterized by first, the facts that any object is meaningful (meaning being linked to perceptual experience, in particular linked to traces left by previous experiences), and second by their practical value, objects are immediately associated to a meaning for action. For instance, according to [3], experts almost never plan, they use their skills, action rules, which require an ability to distinguish perceptual clues. One can almost talk of "situated" routines which rely heavily on what is on hand. In that case, instead of reasoning about the world, one accesses it directly by perception and manipulation. But this notion needs to be extended since the work of [11] has shown that the separation between perception and manipulation (action) is artificial and that our access to the world is more likely done directly by means of an association between a perception and an action through a law that ensures the conformity of this association with the state of the world.

The theory of sensorimotor contingencies has been introduced by [10]. It re-integrates action and physical stimulus into the heart of perception, by postulating that sensations are explained by properties of the sensorimotor dependencies (relationship between motor commands and sensorial feedback) rather than sensorial input only. Following this theory, [11] has shown that with a simplified model of the world, it is possible for an entity to experience alone a sensorimotor law which implies unknown sensorial codes and motor commands, and to extract from this law a set of properties which are independent of the codes and commands, provided it receives enough information about its environment. This properties thus reflect an "objective" property of the environment, like for instance the dimensionality of the physical space. The resulting perception is embodied, since it depends on the sensorimotor experience, but it is independent from any specific motor command or sensorial apparatus.

In computational linguistics [9] as well as in models of language evolution [7], semantic representations are strictly descriptive. All the models deal only with the input of the system they consider. They are passive models of knowledge representation which cannot faithfully convey the intrinsic quality of meaning, since the signification that something has for an entity necessarily entails a part of manipulation. Even if some systems like OPENCYC encode the physical laws of the world in their definitions, a machine using such definition will

never have enough information for all the situation it could face in the real world. For instance if we look at the definition of the verb "to walk", there is no information in the definition about the physical reality of the action in situ. To solve this problem, we propose to integrate the notion of sensorimotor contingencies into semantic representations of the world. In this tripartite model, the sensorimotor contingencies bring the knowledge of the laws regulating the world and the entity, the perception part carries perceptive categories and the action part holds functional categories (objects uses). Contingencies are the mean we propose to use for grounding an entity in its world.

4 Learning to name Actions

With the new types of games we introduce here, we are entering the realm of Distributed Situated and Cognitive Artificial Intelligence. In this sort of game, a population of situated agents moves around in a closed world. When not involved in game playing, agents use their contingency function to constantly update their world representation according to the perception they have of their environment when performing random actions. When agents meet, they try to transmit a piece of their knowledge, by means of a symbol, which can be assimilated by analogy with humans to a verb in imperative form. The knowledge which is the subject of the interaction corresponds to the representation that an agent can build in response to spatiotemporal modifications of its environment following a motion (rotation or translation) of himself or of another agent present in its perceptive field [13]. Of course, to play this game, at least two agents need to have their perceptive field intersected. One agent plays the role of locutor; he randomly selects a word from his memory and enunciates it while computing a prediction of what should be his next perception. If his memory is empty, he invents a new word by combining randomly a fixed number of syllables. The other agent uploads into his working memory all the actions that he find associated in world representation with the word perceived and performs the action that he thinks to be the most appropriate for the current context, thus playing the role of actor. The locutor then gives the result of the game by enunciating the word that corresponds to the action he just perceived. The game is a success if the actor has performed the action expected by the locutor, i.e. the modification of locutor's perception corresponds to the one he expected. In that case, the locutor enunciates twice the same word during the game. Otherwise the locutor, using the same process for action selection as the actor (which involves the working memory) enunciates the word he thinks to be associated with the action he just perceived. Both agents update their world representation according to the game result before restarting their exploration.

In our protocol, a series of games is divided into epochs during which each agent must play at least a given number of times the role of locutor in order to ensure that their knowledge is propagated enough throughout the population. For this game to be possible, we propose the following four hypothesis:

1. All the agent have the same motor and cognitive functionality.
2. The sensorimotor categories corresponding to the different classes of actions (here 2 rotations and 1 translation) must exist BEFORE the learning of their designation. An agent cannot talk about something he has not experienced himself before.
3. As a consequence of the previous hypothesis, the agents must have a categorization function that give them the means to organize the content of their long-term memory and their cognitive operations.

This categorization function must take into account the intrinsic characteristics of the agents and the laws of the world, i.e. the sensorimotor contingencies.

4. In order to validate a game, the locutor must be able to somehow “simulate” the behavior of the actor and his perception if he were performing the action in place of the actor [1].

To be able to verify the third hypothesis, our action designation experiments should have been preceded by an experiment of autonomous learning of sensorimotor contingencies, but since this concerns a rather young field of research, no appropriate software is readily available. Because we know the laws that rule our artificial 2D world, we decided to equip our agents with a priori hand-coded sensorimotor contingencies function. Our aim is not to conduct research in situated perception but to study the emergence of meaning (the only results about sensorimotor contingencies that exist in the domain of situated perception concern the perception of the spatial dimensions and of colors, and they concern a unique agent in a noise-free environment[11]). This little liberty taken with respect to our initial line of conduct, namely to have everything emerge from the autonomous behavior of the agent, does not invalidate our main hypothesis on meaning emergence, since if such meaning does emerge in the population, it would still have been built only from sensorimotor contingencies. But even if our agents have been given the laws that govern the spatial and temporal aspects of their world, they still need to learn their environment, in particular because of its topology, of the objects it could contain, or of the other agents present in it. So when not involved in a designation game, our agents will use their sensorimotor contingencies function to structure the representation of the world they maintain in their long term memory. They do so by choosing an action at random, computing the expected perception, performing the action and comparing the actual perception to the one expected. The recording of the result of an interaction is done according to a similarity function which induces a classification over the event stored in the long term memory, and those categories or subcategories are called “concepts”. Such event record the current action, the current perception and the expected perception in addition to a credit function. Let us note that our model, although quite simple, can generalize easily to any kind of representation of the associations between action and perceptions. We will now look at the experimental results we have obtained with our action designation experiments.

5 Action Naming Experiments

We first present a study of the convergence of a population of agents toward the use of a common set of symbols to name basic actions, then we detail what influence the presence of a working memory in the cognitive structure of the agents has on the lexicon building process. Finally, we show the pertinence of using sensorimotor contingencies in the agent representations for elaborating a common lexicon of action names. In the following series of experiments, we consider a population of 6 agents functioning in a 2D world made of a game board of 8×8 square locations (see figure 1).

In figure 2 we can observe a rapid increase of games successes for the three curves, i.e. independently of the working memory size, but two points are remarkable:

- convergence is not monotonous,
- convergence speed depends on the working memory size.

The first point is explained by the embodiment of the agents, our agents are situated so they are more likely to interact with agents

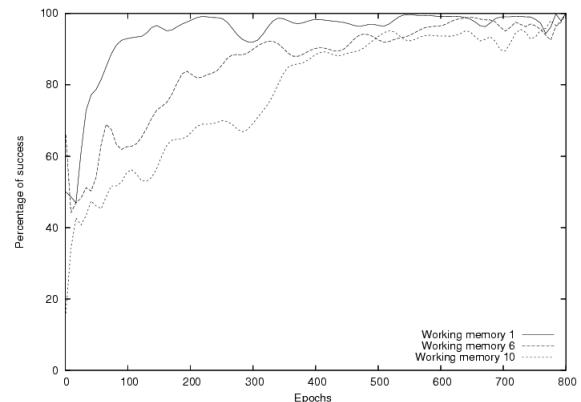


Figure 2. Success rate at action designation in function of time (number of epochs played, up to 800) for 3 different sizes of the working memory (1, 6, and 10).

located in their neighborhood, as a consequence the information flow throughout the population is not evenly distributed but depends on the motion of the agents. The second point stems from an interaction between the working memory size and the reinforcement learning process. Let have a more detailed look at these two phenomena.

5.1 Agent's Embodiment

Because of the situated character of the simulation, there is no certainty that each agent could systematically communicate with all other agents, as opposed to what happens in the more “classical” implementation of our object naming game with disembodied agents. Nevertheless, figure 2 shows convergent systems. Geographically determined subpopulations that share a local lexicon do exist, but most importantly there is also a set of common symbols that is shared by the whole population, as table 1 shows. It contains the number of words shared between all pairs of agents at epoch 800, with agents equipped with a working memory of size 6. All the agents have 14 words in common, but there are some agents with more words in common, a consequence of the locality induced by having embodied agents.

Mixing of information between the subpopulations induced by locality appears in the convergence curves as oscillations in the game success rate. When two subpopulation merge the disagreements between the two lexicons generates failure at the naming game.

agents	1	2	3	4	5	6
1	27	20	23	29	21	
2		25	26	33	25	
3			24	24	19	
4				28	20	
5					26	
6						26

Table 1. This table shows for each pair of agents the number of words they share at epoch 800.

5.2 Working Memory

As we have already said in section 2, to equip the agents with a working memory has the effect of reducing the ambiguity in the lexicon. If we look at figure 2, we observe a very strong levelling of the curve slopes as soon as the working memory is greater than 1. The most striking example is for a working memory of size 1, where we see a sharp progression up to the 200th epoch after which convergence increase diminished drastically.

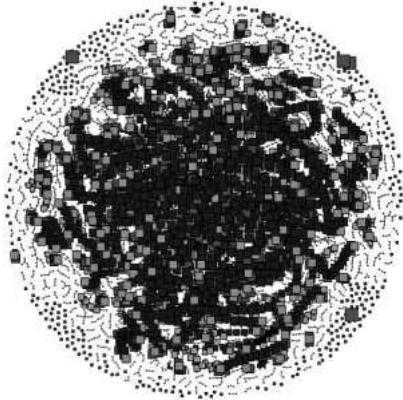


Figure 3. Graph representation of the long term memory of an agent after convergence for a working memory of size 1. Nodes represent concepts (sensorimotor categories) and edges represent inclusion relationships between concepts.

To find an explanation for this fact, we need to look at the internal representation built by the agents, examples of which are shown in graphs³ in figures 3 and 4.

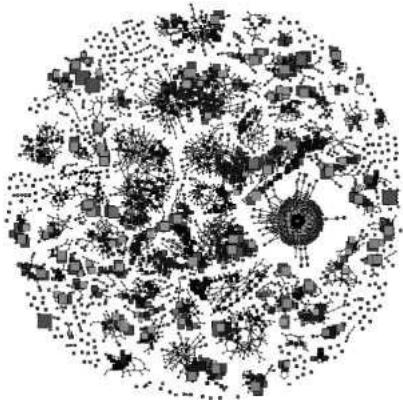


Figure 4. Graph representation of the long term memory of an agent after convergence for a working memory of size 10.

Edge density is linked to working memory size. The smaller the working memory is, the denser the graph is. Density is function of the number of nodes in the graph and so of the number of concepts (categories and subcategories) present in the internal representation of the agents. The working memory is involved in the valida-

tion/validation process of concepts since it contains the most salient concepts in the long term memory of the agent depending on the current situation. A working memory of size 1 does not always permit the selection of all the concepts pertinent for the situation, as a result new concepts will be created while there may exist in the long term memory of the agent concepts that were pertinent but could not be selected because the working memory was already full or because their credit was too low to permit their selection. On the other hand, a working memory of size 10 will select a larger number of concepts at each game and thus entail a reduction of the credit of all the concepts selected if the game is a failure. The greater the working memory size is, the wider the spreading of the penalty will be over the concepts in case of failure. The credit associated to a concept being the ratio between the number of times a concept has been successfully used over the number of times it has been selected (loaded in the working memory), this last number tends to grow along with the working memory size, since selection chances increase with the working memory size. Since penalty is more widespread among the concepts when the working memory is bigger, the saliency of concepts is decreased accordingly and as a consequence also the success rate of the game, because local dialects are preserved longer. In the end, this results corroborates what we have already found in the context of the object naming game described in section 2, the larger the working memory is, the better context disambiguation will be. But contrary to what was observed in the object naming game, here a better disambiguation (larger working memory) slows down convergence toward a common lexicon since it tends to spread penalty over all concepts, including the potentially pertinent concepts.

5.3 Sensorimotor Contingencies

Still with the same action naming game protocol, we will now show the pertinence of using a perception-action-contingency triplets to build and access the world representation stored in the agent's long term memory. We will proceed systematically and consider a population of situated agents which uses different types of information to build their world representation:

- only perceptions,
- only actions,
- perception and actions,
- and finally perception-action-contingency triplets.

Non converging games (perception only or action only): Figure 5 shows non convergence for a system which takes into account only perceptions. Since a key factor in the action naming game is the ability to predict the expected state of the environment following an action (expectation), a system that considers only perception cannot associate together all the perceptions generated by a given action. The agents cannot link causes to effects. The system is then “crushed” by perceptive combinatoric explosion and cannot anymore build sensorimotor categories. The required hypothesis of preexistence of the sensorimotor categories of section 2 for the action naming game to be convergent is not verified anymore. Nevertheless, the curve of figure 5 shows a certain amount of success, but it is only due to the random selection process for action given a perception, once in a while the agent selects by chance the correct action among all the possible ones.

Similarly, if we consider only action as access key for the agent's representation we face a paradox: all the games are successes but the population does not converge toward a common lexicon! This is

³ The graph were created with the Tulip visualizer of LABRI.

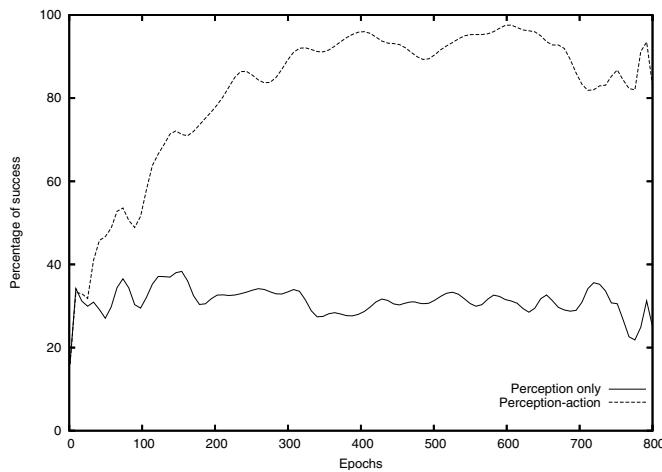


Figure 5. Success rate curve in function of time at naming action game for a population of agents which consider only perceptions and perception-action pairs for its representation (working memory of size 6, 800 epochs).

entirely due to the protocol, since the validation of a game is based on the comparison between an effective perception and an expected perception. If we do not take into account perceptions, all games are successes, necessarily. This degenerated case is of no interest for our purpose.

Converging games (perception-action and contingencies): If now we consider the pair perception-action as access key for the agent's representation (we will call it the PA case), we now obtain the expected convergence, as is shown in figure 5. Similarly, if we consider the triplet perception-action-contingency as access key (PAC case) we get exactly the same kind of convergence as for the perception-action case. So what is the interest of adding contingency information if both approaches result in the same convergence? The difference lies in the quality of the lexicon which improves drastically if we use sensorimotor contingency information. Let us consider, for instance, an agent that tries to perform a forward translation when facing a wall. In the PA case, “consistency” of the phenomena will not be checked and whatever the perceptive outcome of an action, it will serve as a base for building a sensorimotor category, which eventually will fall under the translation category (a particular sub-category of). But in the PAC case, the same situation will end up with the creation of a category different from translation (which we could call “no-action”) since the expected perception (change of location) predicted by the sensorimotor contingencies is not validated for an the agent trying to move into a wall. So adding the sensorimotor contingencies is equivalent to introducing consistency checking in the representation of the agent to ensure it reflects the effective laws of its environment by detecting “anomalies”. The addition of contingency information will result in more sensorimotor categories being created and in larger lexicons. Indeed, we measured experimentally for the size of the lexicon holding the associations between names and sensorimotor categories, an average lexicon size of 4 in the PA case and an average size of 14 in the PAC case.

6 Conclusions

These experiments of unsupervised learning action designation with situated cognitive agents in a 2D environment provided us with a first experimental validation of the possibility for a population of agents to converge toward a common lexicon of basic action names, while relying only on the representation built from sensorimotor contingencies recording and categorization, done when the agent interacts with its environment and other agents. Further, it enabled us to exhibit a necessary condition for the game to converge, namely the ability for an agent to project itself “mentally” in place of another agent in order to predict its perception. Note that in the case of simpler forms of the naming games, involving only static objects and disembodied agents, this condition does not hold. We have also shown that:

1. the situated character of agents generates the emergence of local lexicons (dialects) which tend to hinder global convergence toward a common lexicon
2. also the augmentation of the size of the working memory tends to reduce the ambiguity in the shared lexicon, it has the negative effect of preserving the existence of local dialects, and thus of slowing convergence.

Adding the sensorimotor contingencies to the agent's world representation is equivalent to introducing consistency checking in the representations to ensure that it reflects the effective laws of the environment. The addition of contingency information will result in larger lexicons, since the agents are then able to make finer grain distinction over the consequences of their actions in the environment.

REFERENCES

- [1] M.A. Arbib, Billard, A., Iacoboni M., and Oztop E., ‘Synthetic brain imaging: grasping, mirror neurons and imitation’, *Neural Networks*, **13**, 975–997, (2000).
- [2] A. Baddeley, *Human memory, theory and practice*, Lawrence Erlbaum Associated Ltd Publishers, 1990.
- [3] R. A. Brooks, ‘Intelligence without representation’, *Artificial Intelligence*, **47**, (1991).
- [4] J. Ferber, *Les systèmes multi-agents*, Inter Edition, 1995.
- [5] J. J. Gibson, *The ecological approach to visual perception*, Lawrence Erlbaum Associates, London, 2nd edition (1986) edn., 1979.
- [6] I. M. L. Hunter, *Memory: Facts and Fallacies*, Penguin, Baltimore, 1957.
- [7] F. Kaplan, *Construction sociale du sens*, chapter Approche dynamique de la cognition, 201–217, Traité des sciences cognitives, Hermès Science Publications, 2002.
- [8] S. Kirby, ‘Spontaneous evolution of linguistic structure: an iterated learning model of the emergence of regularity and irregularity’, *IEEE Transactions on Evolutionary Computation*, (2001).
- [9] D. Lenat, ‘Toward programs with common sense’, *Communications of the ACM*, **33**(8), 30–49, (1990).
- [10] J. K. O'Regan, ‘Sensorimotor account of vision and visual consciousness’, *Behavioural and Brain Sciences*, **24**(5), 883–917, (2001).
- [11] D. Philippon, J.K. O'Regan, and J.-P. Nadal, ‘Is there something out there? inferring space from sensorimotor dependencies’, *Neural Computation*, **15**(9), 2029–2050, (2003).
- [12] J. Poudade, ‘Cognitivist lexicon improvement in language games’, in *Proceedings of the International Colloquium “Logic, Games and Philosophie”: Foundational Perspectives*, Prague, (2004).
- [13] J. Poudade, Landwerlin L., and Paroubek P., ‘Des agents cognitifs situés apprennent à désigner des actions’, in *Proceedings of the 15th congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, Tours, (January 2006).
- [14] L. Steels, ‘Self organizing vocabularies’, in *Proceeding of the Vth Alife Conference*, ed., Langton C., Nara, Japan, (1997).

Tracking the Lexical *Zeitgeist* with WordNet and Wikipedia

Tony Veale¹

Abstract. Most new words, or neologisms, bubble beneath the surface of widespread usage for some time, perhaps even years, before gaining acceptance in conventional print dictionaries [1]. A shorter, yet still significant, delay is also evident in the life-cycle of NLP-oriented lexical resources like WordNet [2]. A more topical lexical resource is Wikipedia [3], an open-source community-maintained encyclopedia whose headwords reflect the many new words that gain recognition in a particular linguistic sub-culture. In this paper we describe the principles behind *Zeitgeist*, a system for dynamic lexicon growth that harvests and semantically analyses new lexical forms from Wikipedia, to automatically enrich WordNet as these new word forms are minted. *Zeitgeist* demonstrates good results for composite words that exhibit a complex morphemic structure, such as portmanteau words and formal blends [4, 5].

1 INTRODUCTION

Language is a dynamic landscape in which words are not fixed landmarks, but unstable signposts that switch directions as archaic senses are lost and new, more topical senses, are gained. Frequently, entirely new lexical signposts are added as newly minted word-forms enter the language. Some of these new forms are cut from whole cloth and have their origins in creative writing, movies or games. But many are patchwork creations whose origins can be traced to a blend of existing word forms [1]. This latter form of neologism is of particular interest to the computational lexicographer, since such words possess an obviously compositional structure from which one can begin to infer meaning. In this paper, we demonstrate that, if given enough semantic context, an automated system can assign a sufficiently rich semantic structure to these words to allow them to be automatically added to an electronic database like WordNet [2]. When tied to a system for harvesting new word forms from the internet, this capability allows for a dynamic WordNet that grows itself in response to a changing language and cultural context.

Most neologisms bubble beneath the surface of widespread usage before they gain entry to a conventional dictionary. This is to be expected, since the internet is awash with idiosyncratic neologisms that lack both charm and staying power. Nonetheless, to experience the variety and inventiveness of the most creative new words in English, one need look no further than Wikipedia [3], an open-source electronic encyclopedia that is continuously updated by a on-line community of volunteers. If such words are likely to be encountered in any text to which NLP technologies are applied, from deep text understanding to shallow spell-checking, we should expect our lexical databases to possess a basic interpretation capability.

In this paper, we describe an automated system, called *Zeitgeist*, that harvests neologisms from Wikipedia and uses the semantic context provided by Wikipedias topology of cross-references to add corresponding semantic entries to WordNet. In section two we briefly describe WordNet and Wikipedia, and outline the properties of each that are central to *Zeitgeist*'s operation. Our goal is to exploit only the topology of cross-references, rather than the raw text of the corresponding Wikipedia articles (which would necessitate heavy-duty parsing and analysis methods). Since some topological contexts are more opaque than others, *Zeitgeist* employs a multi-pass approach to acquiring new word forms. In the first pass, only clear-cut cases are harvested; these exemplars are then generalized to underpin schemata that, in a second pass, allow less obvious neologisms to be recognized and semantically analyzed. Both passes are described in sections three and four. In section five, an empirical evaluation and discussion of *Zeitgeist*'s results is presented, while concluding thoughts are offered in section six.

2 LINKING WORDNET AND WIKIPEDIA

WordNet and Wikipedia each blur the traditional semiotic distinction between dictionaries and encyclopedias - which views the former as a source of *word* knowledge and the latter as a source of *world* knowledge - in different ways. WordNet is primarily an electronic dictionary/thesaurus whose structure is informed by psycholinguistic research (e.g., it uses different representations for nouns, verbs, adjectives and adverbs), but in eschewing alphabetic indexing for a semantic organization, it imposes an encyclopedia-like topic organization on its contents. Its coverage is broad, containing entries on topics such as historical events, places and personages more typically found in an encyclopedia. Unsurprisingly, it tends to be used in NLP applications not just as a lexicon, but as a lightweight knowledge-base for reasoning about entities and events.

For its part, Wikipedia's topic articles are surprisingly word-oriented. One finds many more headwords than in a conventional encyclopedia, and a richer level of interconnectedness. In many cases, composite headwords (such as "feminazi") are explicitly linked to the entries for their component parts, while detailed articles on lexical phenomena such as blended (or portmanteau) word-forms [4, 5] and political epithets provide links to numerous topical examples. Additionally, a sister project, Wiktionary [6], aims to exploit the Wikipedia model for an open-source dictionary.

The advantages accruing from an integration of such complementary resources are obvious. To Wikipedia, WordNet can give its explicit semantic backbone, as found in the *isa*-taxonomy used to structure its noun senses. To WordNet, Wikipedia can give its rich, open-textured topology of cross-references [7], as well as its larger and

¹ School of Computer Science and Informatics, University College Dublin, Ireland, email: {Tony.Veale}@UCD.ie

constantly growing set of topical headwords. To achieve this integration, the headwords of Wikipedia must be sense-disambiguated, though [8] report positive results for this task. In this paper, we explore the extent to which the semantic head of a neologism (that part which contributes the suffix, partially or completely, such as “pub” in “Gastropub” and “economics” in “Enconomics”) can be disambiguated by the priming effects of other links emanating from the same Wikipedia article. General purpose WSD techniques (e.g., [9,10]), applied to the text rather than the links of an article, can then be used to resolve those ambiguous heads that are not primed in this way.

Toward this end, we introduce two connectives for relating Wikipedia headwords to WordNet lexical entries. The first is written $x \text{ } isa \text{ } y$, and states that a new synset $\{x\}$ is to be added to WordNet as a hyponym of the appropriate sense of y . Thus, *superhero* *isa* *hero* assumes that WSD is used to identify the intended sense of “hero” in the “superhero” context. The second is $x \text{ } hedges \text{ } y$, as in *spintronics* *hedges* *electronics*. As described in Lakoff [11], a hedge is a category-building relationship that allows one to reason as if a concept belonged to a given category, in spite of strict knowledge to the contrary (e.g., most people know that whales are not fish, but reason about them as if they were). In WordNet terms, hedge relationships will ultimately be instantiated via taxonomic coordination: $\{\text{spintronics}\}$ will not be added as a hyponym of $\{\text{electronics}\}$, rather both will share the common hypernym $\{\text{physics}\}$. Hedges allow us to sidestep the awkward issues of hyperbolae and metaphor that frequently mark new coinages. Though “affluenza” (“affluence + influenza”) is not, strictly speaking, a kind of “influenza”, the hedge allows an NLP system to reason as if it were a real virus; this is apt, since the blend is used to depict affluence as a contagious affliction.

3 PASS I: LEARNING FROM EASY CASES

We employ a string-matching approach to recognizing and analyzing Wikipedia neologisms, in which specific schemata relate the form of a headword to the form of the words that are cross-referenced in the corresponding article. Let $\alpha\beta$ represent the general form of a Wikipedia term, where α and β denote arbitrary prefix and suffix strings that may, or may not, turn out to be actual morphemes. In addition, we use $\alpha \rightarrow \beta$ to denote a reference to headword β from the Wikipedia article of α , and use $\alpha \rightarrow \beta ; \gamma$ to denote a contiguous pair of references to β and γ from article α .

As noted earlier, *Zeitgeist* seeks out neologisms that are a formal blend of two different lexical inputs [4, 5]. The first input contributes a prefix element, while the second contributes a suffix element that is taken to indicate the semantic head of the neologism as a whole. The first schema below illustrates the most common arrangement of lexical inputs (as we shall see in section 5):

Schema I: Explicit Extension

$$\frac{\alpha\beta \rightarrow \beta \wedge \alpha\beta \rightarrow \alpha\gamma}{\alpha\beta \text{ } isa \text{ } \beta}$$

This schema recognizes blended word forms like “gastropub” and “feminazi” in which the suffix β is a complete word in itself (e.g., “pub” and “Nazi”), and in which the prefix α is a fragment of a contextually linked term (like “gastronomy” or “feminist”). The suffix β provides the semantic head of the expansion, allowing the new term to be indexed in WordNet under the appropriate synset (e.g., {Nazi} or {pub, public_house}). The textual gloss given to this new entry

will be a simple unpacking of the blended word: “ $\alpha\gamma\beta$ ” (e.g., “gastro~~onomy~~ pub” and “feminist Nazi”). To avoid degenerate cases, α and β must meet a minimum size requirement (at least 3 characters apiece), though in some exceptional contexts (to be described later), this threshold may be lowered.

Many neologisms are simple variations on existing terminology. Thus, “fangirl” is a male variation on “fanboy”, while “supervillain” is a criminal variation on “superhero”. When an explicit Wikipedia reference exists between these alternating suffixes, the new composite word can be identified as follows:

Schema II: Suffix Alternation

$$\frac{\alpha\beta \rightarrow \alpha\gamma \wedge \beta \rightarrow \gamma}{\alpha\beta \text{ } hedges \text{ } \alpha\gamma}$$

This schema identifies a range of alternating suffix pairs in Wikipedia, from *man* \leftrightarrow *boy* to *woman* \leftrightarrow *girl* to *genus* \leftrightarrow *genera*, *bit* \leftrightarrow *byte* and *bacteria* \leftrightarrow *toxin*.

We can now begin to consider portmanteau words in which the suffix term is only partially present. Words like “Rubbergate” are understood as variations on other terms (e.g., “Watergate”) if the prefix term (here, “rubber”) is explicitly linked. In effect, a partial suffix like “gate” becomes evocative of the whole, as follows:

Schema III: Partial Suffix

$$\frac{\alpha\beta \rightarrow \gamma\beta \wedge (\alpha\beta \rightarrow \alpha \vee \alpha\beta \rightarrow \gamma \rightarrow \alpha)}{\alpha\beta \text{ } hedges \text{ } \gamma\beta}$$

This schema additionally covers situations where the prefix is only indirectly accessible from the neologism, as in the case of “metrosexual” (where “metro” is accessible via a link to “metropolitan”), and “pomosexual” (where “pomo” is only accessible via a mediating link to “postmodernism”). We note that this schema ignores the obvious role of rhyme in the coinage of these neologisms.

This indirection means that, in words like “metrosexual”, both the prefix and the suffix may be partially projected to form a true portmanteau word. In Wikipedia, the lexical inputs to a portmanteau word are often stated as contiguous references in the corresponding article. For instance, Wikipedia describes “sharpedo” as a “shark torpedo”, while “Spanglish” is explicitly unpacked in the corresponding article as “Spanish English”. We can exploit this finding in the following schema:

Schema IV: Consecutive Blends

$$\frac{\alpha\beta \rightarrow \alpha\gamma ; \gamma\beta}{\alpha\beta \text{ } hedges \text{ } \gamma\beta}$$

Indeed, portmanteau terms are so striking that the corresponding Wikipedia articles often explicitly reference the headword “portmanteau”, or vice versa. In such cases, where $\alpha\beta \rightarrow \text{portmanteau}$, we can safely reduce the minimum size requirements on α and β to two characters apiece. This allows *Zeitgeist* to analyze words like “spork” (spoon + fork) and “spongery” (spam + forgery).

4 PASS II: RESOLVING OPAQUE CASES

The foregoing schemata anchor themselves to the local topological context of a headword to curb the wild over-generation that would arise from string decomposition alone. But even when this

topological context is uninformative, or absent entirely (since some Wikipedia articles make no reference to other articles), a system may be able to reason by example from other, more clear-cut cases. For instance, there will be many exemplars arising from schemas III and IV to suggest that a word ending in “ware” is a kind of software and that a word ending in “lish” or “glish” is a kind of English. If E is the set of headwords analyzed using schema III and IV, and S is the corresponding set of partial suffixes, we can exploit these exemplars thus:

Schema V: Suffix Completion

$$\frac{\alpha\beta \rightarrow \gamma\beta \wedge \gamma\beta \in E \wedge \beta \in S}{\alpha\beta \text{ hedges } \gamma\beta}$$

Since the Wikipedia entries for “crippleware”, “donationware” and “malware” - but not “stemware” or “drinkware” - make reference to “software”, the above schema allows us to infer that the former are kinds of software and the latter dishware. Suffix completion reflects the way neologisms are often coined as reactions to other neologisms; for example, once “metrosexual” is recognized using schema III (partial suffix), it provides a basis for later recognizing “retrosexual” using schema V, since “sexual” will now suggest “metrosexual” as a completion. Similarly, “Reaganomics” serves as an exemplar for later analysing “Enronomics”. If P denotes the set of prefix morphemes that are identified via the application of schemas I, II and III, we can also formulate the following generalization:

Schema VI: Separable Suffix

$$\frac{\alpha\beta \rightarrow \beta \wedge \alpha \in P}{\alpha\beta \text{ isa } \beta}$$

This is simply a weakened version of schema I, where α is recognized as a valid prefix but is not anchored to any term in the topological context of the headword.

Though the entry “logicnazi” makes no reference to other headwords in Wikipedia, one can immediately recognize it as similar to “feminazi” (a “feminist Nazi” as resolved by schema I). Conceptually, “Nazi” appears an allowable epithet for an extreme believer of any ideology, and in part, this intuition can be captured by noting that the “Nazi” suffix overwrites the “ism” / “ist” suffix of its modifier. If T is a set of tuples, such as $\langle \text{ism}, \text{Nazi} \rangle$, derived from the use of schema I, we have:

Schema VII: Prefix Completion

$$\frac{\alpha\gamma \rightarrow \alpha \wedge \langle \gamma, \delta\beta \rangle \in T}{\alpha\beta \text{ isa } \beta}$$

Zeitgeist recognizes “logicnazi” as a kind of “Nazi”, in the vein of “feminazi”, since, from “logic” it can reach an “ism” or belief system “logicism” for this Nazi to extol. Likewise, it recognizes “Zionazi” as an extreme Zionist (allowing for a shared “n”), and “Islamonazi” as an extreme Islamist (allowing for an added “o” connective).

Finally, the collected prefixes and suffixes of pass one can now be used to recognize portmanteau words that are not explicitly tagged (as in schema V) or whose lexical inputs are not contiguously referenced (as in schema IV):

Schema VIII: Recombination

$$\frac{\alpha\beta \rightarrow \alpha\gamma \wedge \alpha\beta \rightarrow \delta\beta \wedge \alpha \in P \wedge \beta \in S}{\alpha\beta \text{ hedges } \delta\beta}$$

Thus, a “geonym” can be analyzed as a combination of “geography” and “toponym”.

5 EVALUATION AND DISCUSSION

To evaluate these schemata, each was applied to the set of 152,060 single-term headwords and their inter-article connections in Wikipedia (as downloaded as a SQL loader file in June, 2005). Version 1.6 of WordNet was used to separate known headwords from possible neologisms. In all, 4677 headwords are decomposed by one or more of the given schemata; of these: 1385 (30%) are ignored because the headword already exists in WordNet, 884 (19%) are ignored because the hypernym or hedge determined by the analysis does not itself denote a WordNet term. Thus, though “bioprospecting” is correctly analyzed as “biology prospecting”, “prospecting” is not a lexical entry in WN1.6 and so this term must be ignored. The remaining 2408 (51%) of cases² are analyzed according to the breakdown of Table I:

Table 1. Breakdown of performance by individual schema.

Schema	# Headwords	# Errors	Precision
I	710	29%	.985
II	144	5%	1.0
III	330	13%	.985
IV	82	3%	.975
V	161	6%	1.0
VI	321	13%	.95
VII	340	14%	.90
VIII	320	13%	.965

Each *Zeitgeist* analysis was manually checked to find errors of decomposition and provide the precision scores of Table I. Two schemas (II in pass one, which e.g., derives Rubbergate from Watergate, and V in pass two, which e.g., derives retrosexual from metrosexual) produce no errors, while the most productive schema (explicit extension, schema I) has an error rate of just 1.5%. In contrast, schema VII (prefix completion in pass two, which derives logicnazi via the exemplar feminist/feminazi) is cause for concern with an error rate of 10%. High-risk schemata like this should thus be used in a controlled manner: they should not update the lexicon without user approval, but may be used to hypothesize interpretations in contexts that are more ephemeral and where more information may be available (e.g., a spellchecking or thesaurus application invoked within a particular document).

Some obvious factors contribute to an overall error rate of 4%. Company names (like Lucasfilm) comprise 12% of the erroneous

² Interestingly, the distribution for WN2.1 is much the same: 1570 analysed headwords (33%) are ignored because the headword is already in WN2.1, while 789 headwords (17%) must be ignored because their semantic heads are *not* in WN2.1. This leaves 2319 valid neologisms (49%) to be added to WN2.1, as opposed to 2408 for WN1.6. The number of neologisms remains relatively stable across WN versions because greater lexical coverage presents a greater opportunity to recognize neologisms that cannot be integrated into lesser versions. For instance, the “cyberpunk” entry in WN2.1 means that while this word is not treated as a neologism for this version (as it is for WN1.6), its presence allows “steampunk” and “clockpunk” to be recognized as neologisms.

cases, organization names (like Greenpeace and Aerosmith) 6%, place names (like Darfur) 11% and product names (like Winamp) 2%. Another 5% are names from fantasy literature (like Saruman and Octopussy). In all then, 35% of errors might be filtered in advance via the use of a reliable named-entity recognizer.

5.1 Word Sense Disambiguation

For 51% of the Wikipedia neologisms recognized by *Zeitgeist*, the semantic head (i.e., the word that contributes the suffix to the neologism) denotes an unambiguous WordNet term. The remaining 49% of cases thus require some form of WSD to determine the appropriate sense, or senses, of the semantic head before the neologism can be added to WordNet. While one can employ general purpose WSD techniques on the textual content of a Wikipedia article [9, 10], the topological context of the headword in Wikipedia may, to a certain degree, be self-disambiguating via a system of mutual priming.

For example, the intended WordNet sense of “hero” in the headword “superhero” (not present in WN 1.6) is suggested by the link *superhero* → *Hercules*, since both “hero” and “Hercules” have senses that share the immediate WordNet hypernym {Mythological-Character}. In general, a given sense of the semantic head will be primed by any Wikipedia term linked to the neologism that has a WordNet sense to which the head relates via synonymy, hyponymy or hypernymy.

Priming can also be effected via an intersection of the textual glosses of WordNet senses and the topological context of the Wikipedia article (in a simple Wikipedia variation of the Lesk algorithm [9]). For example, the Wikipedia headword “kickboxing” suggests the ambiguous “boxing” as a semantic head (via schema I). However, because the Wikipedia link *kickboxing* → *fist* is echoed in the gloss of the WordNet sense {boxing, pugilism, fisticuffs} but not in the gloss of {boxing, packing}, only the former is taken as the intended sense.

More generally, the elements of the Wikipedia topological context can be viewed as a simple system of semantic features, in which e.g., *fist* is a feature of *kickboxing*, *fascism* is a feature of *Nazi*, and so on. Furthermore, because blending theory [4,5] claims that blended structures will contain a selective projection of elements from multiple inputs, this projection can be seen in the sharing of semantic features (that is, topological links) between the neological headword and its semantic head. For instance, the Wikipedia terms “Feminazi” and its semantic head, “Nazi”, share three Wikipedia links - to Totalitarianism, Fascism and Nazism - which may be taken as the contribution of the lexical component “Nazi” to the meaning of the word as a whole. In the terminology of blending theory [4,5], these features are *projected* from the input space of *Nazi* into the blended space of *Feminazi*. Projection of this kind occurs in 64neologisms recognized by *Zeitgeist*.

By understanding the projective basis of a word blend, *Zeitgeist* has yet another means of performing disambiguation of the semantic head, since the intended sense of the head will be that sense that visibly contributes semantic features to the blend. In the case of “kickboxing”, the feature *fist* is directly contributed by the pugilistic sense of “boxing”. However, for the blended word “emoticon”, the feature *pictogram* is indirectly contributed by the user-interface sense of “icon” via its hypernym {symbol}.

Overall, topological priming resolves 25% of neologisms to a single WN1.6 sense, while another 1% are resolved to multiple WN senses, which is to be expected when the head element is a polysemous word. For instance, “photophone” (“photograph” + “tele-

phone”) is deemed to hedge both the equipment and medium senses of “telephone”, while “subvertising” (“subversion” + “advertising”) is deemed to hedge the message and industry senses of “advertising”. In all, total WSD coverage in *Zeitgeist* is 77%. Recourse to more general WSD techniques is thus needed for just 23% of cases.

5.2 Literal Versus Figurative Interpretations

Our evaluation reveals that over half (57%) of the neologisms recognized by *Zeitgeist* (via schemas I, VI and VII) are realized in WordNet via a simple hypernymy relationship, while the remainder (43%) are realized (via schemas II, III, IV, V and VII) using the more nuanced *hedge* relationship. It seems clear, for instance, that “Gastropub” really is a kind of “pub” and “cocawine” really is a kind of “wine” (with added cocaine). However, it is not so clear whether Feminazis are truly Nazis (in the strict, National Socialist sense), so hedging may be more prevalent than these figures suggest. Though WordNet defines {Nazi} as a hyponym of {fascist}, the word is often used as a highly charged pseudo-synonym of the latter. “Nazi” seems to be used here in a sense-extensive, metaphoric fashion to suggest totalitarian zeal rather than political affiliation.

Two factors alert us that this use of “Nazi” is hyperbole rather than literal extension. The first is the orthographic form of the word itself, for while “Nazi” is a proper-named class, “Feminazi” employs the word in an uncapitalized form which suggests a process of semantic bleaching or generalization. The second factor is the relative contribution, in terms of projected features, of the semantic head to the blend as a whole. Recall that the word “Nazi” shares the Wikipedia linkages {Totalitarianism, Fascism, Nazism} with “Feminazi”, so these features may be said to originate from this input. However, “fascist” also references the terms {Totalitarianism, Fascism, Nazism} in Wikipedia, suggesting that there is no obvious loss of semantic import if Feminazi is considered an extension of {fascist} rather than of {Nazi}.

In 36% percent of neologisms, one or more semantic features are projected into the blend by a hypernym of the semantic head. In just 2% of neologisms this projection occurs in the context of an *isa* relation (i.e., via schemas I and VI) and is such that all features that are projected from the head are also redundantly projected from the hypernym of the head. (As it happens, only in the case of “Feminazi” does the semantic head denote a proper-named concept). While not conclusive, such redundancy is sufficient cause either to hedge the relationship or to prompt for human guidance in these cases.

6 CONCLUSIONS

We have presented a linguistics-lite approach to harvesting neologisms from Wikipedia and adding them to WordNet. *Zeitgeist* does not employ an explicit morphological analyser, but relies instead on a marriage of partial string-matching and topological constraints. Nonetheless, many of the words that are successfully recognized exhibit a creative and playful use of English morphology. Furthermore, by grounding analyses in the local link topology of Wikipedia articles, *Zeitgeist* gains a semantic insight that one cannot obtain from morphology rules alone. For instance, not only is “microsurgery” recognized as a micro-variant of surgery, the specific meaning of “micro” in this context is localized to the headword “microscopy” via schema I. The concept “microsurgery” is not just “micro-surgery”, but surgery conducted via a microscope.

Even a lightweight approach can, however, bring some degree of semantic insight to bear on the analysis of new words. In this

respect, Wikipedias link topology deserves further consideration as a source of semantic features. Certainly, Wikipedia has great promise as a semi-structured semantic representation. For instance, one can distinguish two kinds of semantic feature in Wikipedia. Strong or highly-salient features are those that are reciprocated; thus, *charity*→*altruism* and *altruism*→*charity* implies that altruism is a highly salient feature of charity, and vice versa. Weak features are those that are not reciprocated in this way. It remains to be seen how far one can go with such a representation without imposing a more rigid logical framework, but we believe that the initial foray described here suggests the scheme has yet more mileage to offer.

We conclude by noting that the linguistics-lite nature of *Zeitgeists* approach means that is not intrinsically biased toward English. In principle, its mix of string matching and topological constraints should validly apply to other languages also. Whether phenomena like lexical blending spring forth with equal regularity in the non-English languages supported by Wikipedia is a subject of future research.

REFERENCES

- [1] Dent, S., Fanboys and Overdogs: The Language Report III. Oxford University Press (2003).
- [2] Miller, G. A., WordNet: A Lexical Database for English. *Communications of the ACM*, Vol. 38 No. 11 (1995).
- [3] www.wikipedia.org
- [4] Fauconnier, G., Turner, M., Conceptual Integration Networks. *Cognitive Science*, 22(2), pp 133187 (1998).
- [5] Veale, T., O'Donoghue, D., Computation and Blending. *Cognitive Linguistics*, 11(3-4).
- [6] www.wiktionary.org
- [7] Ruiz-Casado, M., Alfonseca, E., Castells, P., Automatic Extraction of Semantic Relationships for WordNet by Means of Pattern Learning from Wikipedia. *LNAI 3513*, pp 67 (2005).
- [8] Ruiz-Casado, M., Alfonseca, E., Castells, P., Automatic Assignment of Wikipedia Encyclopedic Entries to WordNet Synsets. *Springer LNAI 3528*, pp 280 (2005).
- [9] M. Lesk, Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *Proceedings of ACM SigDoc Conference, Toronto: ACM*, pp 24-6 (1986).
- [10] P. Resnik, Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research 11*, pp 95-130, (1999).
- [11] G. Lakoff: *Women, Fire and Dangerous Things: How the Mind forms Categories*. University of Chicago Press (1987)

2. Constraints and Search

This page intentionally left blank

Enhancing constraints manipulation in semiring-based formalisms¹

Stefano Bistarelli² and Fabio Gadducci³

Abstract. Many “semiring-like” structures are used in Soft Constraint Satisfaction Problems (SCSPs). We review a few properties of semirings that are useful for dealing with soft constraints, highlighting the differences between alternative proposals in the literature.

We then extend the semiring structure by adding the notion of *division* as a weak inverse operation of product. In particular, division is needed to apply constraint relaxation when the product operation of the semiring is not idempotent. The division operator is introduced via *residuation* and it is also able to deal with partial orders, generalizing the approach given for Valued CSPs.

1 Introduction

Several formalizations of the concept of *soft constraints* are currently proposed in the literature. In general terms, a soft constraint may be seen as a function associating to each assignment (i.e., instantiation of the variables occurring in it) a value in a partially ordered set A , which can be interpreted as a set of preference values or costs. Combining constraints then has to take into account such additional values, and thus the formalism must provide suitable operations for the combination and the comparison of tuples of values and constraints.

The paper focuses on semiring-based frameworks: Assignments take values in a semiring, the order is the one associated to the $+$ operator of the semiring, and the combination is the \times operator. We review the basics of semiring theory and identify a few properties that are needed to deal with constraints. These properties boil down to the notion of commutative, absorptive semiring, an instance of well-known tropical semirings [23]. Based on this characterization, a comparison between several proposals is performed, namely valuation structures [26], c -semirings [4, 7] and semiring valuations [27].

Soft constraint satisfaction problems can be solved by extending and adapting the techniques used for their classical version. For example, a branch and bound search algorithm can be used, instead of backtracking, for finding the best solution. A pivotal component of this generalization is the search for algorithms of constraint relaxation, in order to obtain some form of local consistency (i.e., such that changing the semiring values associated to an assignment does not change the overall solution). More to the point, in this paper we define a technique to move valuation (cost) information from constraints involving multiple variables to simpler, possible unary ones, resulting in an upper bound on the problem valuation.

Elaborating on a proposal by Cooper and Schiex [11, 12, 25], we propose improvements on classical local consistency techniques, in order to apply the framework also whenever the constraint composi-

tion operator is not idempotent. To this end, we extend the semiring structure by adding the notion of *division* as a weak inverse operation of product. The solution we pursue for characterizing that operator is based on *residuation theory* [8], a standard tool on so-called *tropical* arithmetics. It allows for obtaining a division operator *via* an approximate solution to the equation $b \times x = a$.

2 On semirings

Slightly different presentation for the notion of *semiring* occur in the literature. The less constrained definition we are aware of (compare e.g. the survey [24]) is given by a set A equipped with two binary operators, the sum $+$ and the product \times , such that $+$ is associative and commutative (that is, the pair $\langle A, + \rangle$ is a commutative semigroup), and the product operator \times distributes over $+$.

In soft constraints literature the cost/preference associated to each variable instance is modeled as an element of a semiring, and the constraint combination is defined via the semiring product operator [4, 5, 7, 27]. Most often, the sum operator is used just for the induced pseudo order⁴, given by $a \leq b$ iff $a + b = b$.

Which properties should be required for constraint combination?

- Since satisfaction problems are defined by “sets” of constraints, the order of the constraints combination has to be irrelevant. This leads to the *associativity* and *commutativity* for the \times operator;
- since adding constraints decreases the number (and the quality) of the solutions, the combination of constraints has to worsen the value of the operands. This means that the ordering has to be *absorptive* ($a + (a \times b) = a + (b \times a) = a$);
- when dealing with soft constraints there might be the need of representing crisp features. That is, there must be an element in the semiring that has the crisp meaning of total dislike of any solution that involve a specific assignment. Thus, an element $\mathbf{0} \in A$ called *zero* or *annihilator* element ($a \times \mathbf{0} = \mathbf{0} \times a = \mathbf{0}$);
- similarly, there must be an element that has the crisp meaning of “indifference”, i.e., the satisfaction of the specified constraint does not change the overall level of preference for a given tuple. Such an element $\mathbf{1} \in A$ is called *unit* ($a \times \mathbf{1} = \mathbf{1} \times a = a$).

We adopt a terminology inspired by [14] and, in lesser degree, by [24], aiming at a smooth presentation of the main concepts to the reader.

Definition 1 (semirings) A commutative semiring is a five-tuple $\mathcal{K} = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that A is a set, $\mathbf{1}, \mathbf{0} \in A$, and $+, \times : A \times A \rightarrow A$ are binary operators making the triples $\langle A, +, \mathbf{0} \rangle$ and $\langle A, \times, \mathbf{1} \rangle$ commutative monoids (semigroups with identity), satisfying

(**distributivity**) $\forall a, b, c \in A. a \times (b + c) = (a \times b) + (a \times c)$;
 (**annihilator**) $\forall a \in A. a \times \mathbf{0} = \mathbf{0}$.

¹ Research partially supported by the EU IST 2004-16004 SENSORIA and by the MIUR PRIN 2005-015491.

² Dipartimento di Scienze, Università “G. d’Annunzio”, sede di Pescara, bista@sci.unich.it and IIT, CNR, Pisa, Stefano.Bistarelli@iit.cnr.it

³ Dipartimento di Informatica, Università di Pisa, gadducci@di.unipi.it

⁴ A pseudo order is a transitive, antisymmetric, possibly not reflexive relation.

Proposition 1 (absorptive semirings [24, Corollary 2.1]) Let \mathcal{K} be a commutative semiring. Then, these properties are equivalent

(absorptiveness) $\forall a, b \in A. a + (a \times b) = a$,

(top element) $\forall a \in A. a + 1 = 1$.

Semirings verifying the above properties are known as *absorptive* (or *simple*) [10] and represent the structure we put at the base of our proposal since they satisfy the properties absorptiveness, zero and unit element that seem pivotal for any soft constraint framework.

We can now state a simple characterization result linking absorptiveness to idempotency and to the notion of top element.

Proposition 2 (tropical semirings) Let \mathcal{K} be a commutative semiring. If \mathcal{K} is absorptive, then the sum operator is idempotent.

The former result is well-known in the literature, and commutative semirings such that the sum operator is idempotent are called *diodoids* or *tropical semirings*⁵. These structures are well-studied in the literature [1, 2, 23, 24], and we take advantage of some classical constructions in the following sections. Notice in fact that absorptive semirings are just tropical semirings with top element.

2.1 Alternative approaches

We now have a look at some order structures proposed in the literature for interpreting constraints, showing their similarity and highlighting absorptive semirings as a common algebraic structure. More precisely, the section provides a comparison with *c-semiring* [4, 6, 7], valuation structures [26], and semiring valuations [27].

2.1.1 c-semirings

The starting point of our excursus are *c-semirings* [4, 7]: according to our notation, they are tropical semirings with top element. Our analysis, as summed up in Proposition 2, suggests that the intrinsic quality of *c-semirings* is the absorptiveness of the order, whilst idempotency of the sum is merely a side-effect, in the sense that it is implied by the other properties (see Proposition 2).

Note also that most results on local consistency for *c-semirings* require the idempotency of the \times operator [4]. This assumption results in a stronger structure, since $a \times b$ coincides with the greatest lower bound of a and b . Indeed, the focus of this article is also the extension of some algorithms proposed in the *c-semirings* formalism, in order to deal also with non-idempotent product operators.

2.1.2 Valuation structures

Adopting our terminology, a *valuation structure* [26] is a five-tuple $\langle A, \leq, \times, \mathbf{0}, \mathbf{1} \rangle$ such that $\langle A, \times, \mathbf{1} \rangle$ is a commutative monoid, $\langle A, \leq \rangle$ is a total order (with $\mathbf{0}$ and $\mathbf{1}$ as minimum and maximum, respectively), \times is monotonic and $\mathbf{0}$ is its annihilator element.

As noted in [5], a commutative semiring can be associated to each valuation structure by defining the sum operator as $a+b = \max\{a, b\}$, obtaining an absorptive semiring. Moreover, the order in the resulting semiring coincides with the original order of the valuation structure. In fact, note that in a tropical semiring the induced ordering \leq is total iff $a+b \in \{a, b\}$ for all $a, b \in A$, so that there is a one-to-one correspondence between valuation structures and those tropical semirings (known in the literature as *additively extremal* semirings [14, p.11]) such that the associated order \leq is total. We further discuss the properties of these structures later on, when comparing our notion of inverse for the product operator with the proposal in [12].

⁵ The adjective “tropical” was coined by French mathematicians [23] in honor of the Brazilian Imre Simon. The terminology “diodoid” is adopted in [15, 17] to highlight that the structure can not be a ring, but it is “twice a monoid”.

2.1.3 Semiring valuations

Semiring valuations [27] are constraint satisfaction problems taking values in a commutative semiring, where the ordering is the transitive relation $a \leq' b$ iff $\exists c. a + c = b$. The two alternative definitions of orderings coincide for tropical semirings, in the sense that $a \leq b$ iff $a \leq' b$ for all idempotent elements a, b .

The lack of idempotency for the sum operator results in a weaker structure than absorptive semirings, that has proved useful whenever counting the number of solutions is of interest, as for special computations in Bayesian networks. However, the associated reflexive and transitive relation \leq' satisfies relatively few properties, since adding constraints does not lead to worsen the solution, thus resulting in a non-monotonic framework (because of the absence of the absorptive property). These remarks are summed up by the result below.

Proposition 3 Let \mathcal{K} be a commutative semiring. If \leq' is a partial order and $\forall a, b \in A. a \times b \leq' a$, then \mathcal{K} is absorptive.

Semirings such that the relation \leq' is a partial order are known in the literature as *uniquely difference ordered* [14, Section 2]. For these structures, whenever the \times operator worsens the solution, the sum operator is idempotent and thus \leq' equals \leq .

3 Adding division

The search for optimal solutions on constraint satisfaction problems has been mostly based on the idempotency of the \times operator. While the resulting heuristics may have a neat presentation and behavior, many relevant examples fall outside its scope. As we shall see, the viability of local consistency algorithms can be recovered by requiring the existence of a suitable inverse operator.

Among the possible solutions for ensuring the relevant structure, *symmetrisation* embeds the semiring in a larger structure containing an inverse for each element. The approach is quite standard in tropical arithmetics (see e.g. [2, Section 3.4.1.1] and [13, Section 3.8]): the derived semiring has pairs (a, b) as elements, accordingly derived operators, and a minus operator, defined as $-(a, b) = (b, a)$. A related approach is suggested in [11] for obtaining an inverse operator for \times , starting from a *strictly monotonic* valuation structure (i.e., such that $\forall a, b, c \in A. (a < b) \wedge (c \neq \mathbf{0}) \implies a \times c < b \times c$).

These constructions are similar and the properties of the derived inverse operator basically hold due to the totality of the order on valuation structures. This proposal is used in [12, Example 2] as a mechanism for recovering what the authors call *fair* evaluation structures.

The solution we pursue here is based on *residuation theory* [8], a standard tool on tropical arithmetics (see e.g. [2, Section 4.4.2] and [14, Chapter 4]), which allows for obtaining a division operator via an approximate solution to the equation $b \times x = a$. Differently with respect to the use of a completion procedure, no new element is added, leaving the same set of preferences to the user. It is noteworthy that by using the newly defined division operators, suitable consistency algorithms are devised also for non idempotent products.

3.1 Basic definitions and results

This section introduces our notion of invertibility for absorptive semirings: three alternative properties are identified, and they are listed below in the same order of their strength.

Definition 2 Let \mathcal{K} be an absorptive semiring. Then

- \mathcal{K} is invertible if there exists element $c \in A$ such that $b \times c = a$ for all elements $a, b \in A$ such that $a \leq b$;
- it is weakly uniquely invertible if c is unique whenever $a < b$;
- it is uniquely invertible if c is unique whenever $b \neq \mathbf{0}$.

Note that the former definitions do not require the existence for each element a of an inverse, i.e., of an element a^{-1} verifying $a \times a^{-1} = \mathbf{1}$. In fact, the absorptiveness condition $a \times b \leq a$ guarantees that no element, except $\mathbf{1}$ itself, has an inverse.

We now present two results that establish conditions for ensuring invertibility of an absorptive semiring. The first proposition concerns multiplicative idempotent semirings and their order structure.

Proposition 4 *Let \mathcal{K} be an absorptive, multiplicative idempotent semiring. Then, \mathcal{K} is invertible. Moreover, if \leq is a total order, then \mathcal{K} is weakly uniquely invertible.*

The second proposition is for cancellative semirings, i.e., such that $\forall a, b, c \in A. (a \times c = b \times c) \wedge (c \neq \mathbf{0}) \implies a = b$.

Proposition 5 *Let \mathcal{K} be an absorptive, invertible semiring. Then, \mathcal{K} is uniquely invertible iff it is cancellative.*

3.2 On division on residuated semirings

Residuation theory is concerned with the study of sub-solutions of the equation $b \times x = a$. Note that on tropical semirings the relaxed equation $b \times x \leq a$ always has a solution (it is enough to set x to $\mathbf{0}$).

Definition 3 *Let \mathcal{K} be a tropical semiring. Then, \mathcal{K} is residuated if the set $\{x \in A \mid b \times x \leq a\}$ admits a maximum for all elements $a, b \in A$, denoted $a \div b$.*

Note that the previous definition actually suggests an algorithmic heuristics for approximating such a maximal element, intuitively given by the (possibly infinite) sum of all the elements satisfying the inequation [2]. The key point is that the set of sub-solutions of an equation contains also the possible solutions, whenever they exist, and in that case the maximal element is also a solution.

More properties hold if the semiring is absorptive, such as that $b \leq a \implies a \div b = \mathbf{1}$. This fact leads to our notion of invertibility.

Definition 4 *Let \mathcal{K} be an absorptive, invertible semiring. Then, \mathcal{K} is invertible by residuation if the set $\{x \in A \mid b \times x = a\}$ admits a maximum for all elements $a, b \in A$ such that $a \leq b$.*

With an abuse of notation, the maximal element among solutions is denoted $a \div b$. This choice is not ambiguous: if an absorptive semiring is invertible and residuated, then it is also invertible by residuation, and the two definitions yield the same value.

3.3 On complete semirings

Being residuated is implied by a sometimes easier to check property, namely, the existence of elements representing infinite sums.

Definition 5 *Let \mathcal{K} be a tropical semiring. Then, \mathcal{K} is complete if it is closed with respect to infinite sums, and the distributivity law holds also for an infinite number of summands.*

Also associativity and commutativity need to be generalized, and we refer the reader to e.g. [14, Section 3].

Proposition 6 ([2, Theorem 4.50]) *Let \mathcal{K} be a tropical semiring. If \mathcal{K} is complete, then it is residuated.*

The above proposition ensures that all classical soft constraint instances (see Section 4.1) are residuated (because complete) and the notion of division can be applied to all of them.

3.4 Further comparison with valuation structures

We now compare our definition of division with the proposal by Cooper and Schiex in [12, Definition 3.1]. A valuation structure is called *fair* if the set $\{x \in A \mid b \times x = a\}$ has a minimum whenever $a \leq b$ [12, Definition 3.1]. Hence, the division operator is defined as

$$a \div' b = \begin{cases} \min\{x \mid b \times x = a\} & \text{if } a \leq b, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Let us say that an absorptive semiring is *fair* if it is invertible and the operator \div' above is well defined. How does fairness compare with residuation? By definition $\{x \mid b \times x = a\} \subseteq \{y \mid b \times y \leq a\}$, so that the operation $a \div' b$ returns a smaller value than $a \div b$. Nevertheless, the two notions sometimes coincides, e.g. if a semiring is uniquely invertible, hence whenever the \times operator is cancellative (equivalently, whenever a valuation structure is strictly monotonic).

Proposition 7 (cancellativeness) *Let \mathcal{K} be an absorptive, cancellative semiring. Then, \mathcal{K} is fair iff it is invertible by residuation, and in such a case $a \div' b = a \div b$ for all elements a, b such that $a \leq b$.*

A weakly uniquely invertible semiring is invertible by residuation but it is not necessarily fair: the operation $a \div' a$ might not be defined, while $a \div a = \mathbf{1}$ always holds. In fact, the two operators usually differ for the division of an element by itself, since e.g. $a \div' a = a$ holds for multiplicative idempotent semirings. Combined with Proposition 4, relating weakly invertibility and multiplicative idempotency, the remark above establishes the correspondence result below.

Proposition 8 (idempotency) *Let \mathcal{K} be an absorptive, multiplicative idempotent semiring. Then, \mathcal{K} is fair. Moreover, if \leq is a total order, then \mathcal{K} is invertible by residuation, and in such a case $a \div b = a \div' b = a$ for all elements a, b such that $a < b$.*

For those semirings where \leq is total, the proposition also yields that $b \times (a \div' b) = b \times (a \div b) = a$ for all a, b such that $a \leq b$.

We provide no general correspondence result between the two properties. Note however that by definition a residuated fair semiring is invertible by residuation, hence the proposition below follows.

Proposition 9 (completeness) *Let \mathcal{K} be an absorptive, complete semiring. If \mathcal{K} is fair, then it is invertible by residuation.*

In general, the two operations returns different values, as it happens for the set-based CSPs presented in Section 4.1.4. Note that most case studies in the literature satisfy the completeness property, and are either multiplicative idempotent or cancellative: this fact hold for all the examples presented in Section 4.1.

4 Soft constraints and local consistency

Several formalizations of *soft constraints* are currently available. The first part of this section briefly introduces the semiring-based formalism, directly borrowing from the *c-semirings* approach [4, 7]. The second part of this section represents another technical contribution of our work: It presents an extension of local consistency techniques for invertible semirings, thus generalizing those previously proposed for those cases where the \times operator is idempotent.

4.1 Constraint problems

Let $\mathcal{K} = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ be an absorptive semiring; V a finite (possibly ordered) set of variables; and D a chosen domain of interpretation for V . Then, a *constraint* $(V \rightarrow D) \rightarrow A$ is a function associating a value in A to each assignment $\eta : V \rightarrow D$ of the variables.

We define \mathcal{C} as the set of constraints that can be built starting from \mathcal{K} , V and D . Note that even if a constraint involves all the variables in V , it must depend on the assignment of a finite subset of them. For instance, a binary constraint $c_{x,y}$ over variables x, y is a function $c_{x,y} : (V \rightarrow D) \rightarrow A$ which depends only on the assignment of variables $\{x, y\} \subseteq V$. We call this subset the *support* of the constraint. Often, if V is ordered, an assignment (over a support of cardinality k) is concisely presented by a tuple in D^k .

More formally, let $c \in \mathcal{C}$ be a constraint. We define its support as $\text{supp}(c) = \{v \in V \mid \exists \eta, d_1, d_2. c\eta[v := d_1] \neq c\eta[v := d_2]\}$, where

$$\eta[v := d]v' = \begin{cases} d & \text{if } v = v' \\ \eta v' & \text{otherwise} \end{cases}$$

While $c\eta$ is the application of a constraint function $c : (V \rightarrow D) \rightarrow A$ to a function $\eta : V \rightarrow D$, obtaining a semiring value, $c\eta[v := d]$ means $c\eta'$ where η' is η modified with the assignment $v := d$.

Later on we implicitly restrict to finitely supported constraints.

4.1.1 Classical CSPs

Classical (crisp) satisfaction problems may be recast to deal with soft constraints by considering the semiring $\mathcal{K}_{CSP} = \langle \{0, 1\}, \vee, \wedge, 0, 1 \rangle$.

The semiring is finite, hence complete (and residuated by Proposition 6). Moreover, the \times operator is idempotent, then the semiring is invertible, and since the order is total, it is weakly uniquely invertible (by Proposition 4). By instantiating the definition of division we get

$$a \div b = \max\{x \mid b \wedge x \leq a\} = (b \implies a)$$

where \implies is the logic implication.

4.1.2 Fuzzy CSPs

Fuzzy CSPs (FCSPs) extend the standard notion by allowing non-crispness features, and can be modeled by the semiring $\mathcal{K}_{FCSP} = \langle [0, 1], \max, \min, 0, 1 \rangle$.

The semiring is clearly complete (and residuated by Proposition 6). Moreover, the \times operator is idempotent, then the semiring is invertible, and since the order is total, it is weakly uniquely invertible (by Proposition 4). By instantiating the definition of division we obtain

$$a \div b = \max\{x \mid \min\{b, x\} \leq a\} = \begin{cases} 1 & \text{if } b \leq a \\ a & \text{if } a < b \end{cases}$$

4.1.3 Weighted CSPs

While fuzzy CSPs associate a level of preference, in weighted CSPs (WCSPs) tuples come with an associated cost to be minimized. The associated semiring structure is in this case $\mathcal{K}_{WCSP} = \langle \mathbb{R}^+ \cup \{\infty\}, \min, \hat{+}, \infty, 0 \rangle$ for $\hat{+}$ the sum of reals.

The semiring is clearly complete (and residuated by Proposition 6). The \times operator is not idempotent, but the semiring is nevertheless invertible, as it can be easily proved by checking the definition itself. Moreover, the semiring is cancellative, hence it is uniquely invertible by Proposition 5. By instantiating the definition of division we obtain

$$a \div b = \min\{x \mid b \hat{+} x \geq a\} = \begin{cases} 0 & \text{if } b \geq a \\ a \hat{-} b & \text{if } a > b \end{cases}$$

where $\hat{-}$ is the arithmetic difference of a and b .

4.1.4 Set-based CSPs

An interesting class of instances of the soft constraint framework is based on set operations like union and intersection, using the semiring $\mathcal{K}_{set} = \langle \wp(A), \cup, \cap, \emptyset, A \rangle$, where A is any set: the order $\leq_{\mathcal{K}_{set}}$ reduces to set inclusion, and therefore it is partial.

The semiring is clearly complete (and residuated by Proposition 6). The \times operator is idempotent, then it is invertible; but the order is not total, so that it is not uniquely invertible (Proposition 4 can not be applied). By instantiating the definition of division we obtain

$$a \div b = \bigcup\{x \mid b \cap x \subseteq a\} = (A \setminus b) \cup a$$

where \setminus is the set difference operator.

4.2 Combining and projecting soft constraints

The *combination* function $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$. Thus, combining two constraints means building a new constraint whose support involves all the variables of the original pairs, and which associates with each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples. Let $c \in \mathcal{C}$ be a constraint and $v \in V$ a variable. The *projection* of c over $V - \{v\}$, written $c \Downarrow_v$, is the constraint c' such that $c'\eta = \sum_{d \in D} c\eta[v := d]$. Informally, projecting means eliminating some variables from the support.

4.3 Local consistency

The main point in the generalization of local consistency techniques to soft CSPs concerns the fact that, instead of removing tuples, local consistency means changing the semiring values associated to them. In particular, the change always brings these values towards the worst element of the semiring, that is, 0 .

Arc-consistency [3, 16] (AC) is an instance of local consistency where the information present in the constraints is propagated over the variables. In fact, an arc-consistency rule considers a constraint, say with support over variables x, y_1, \dots, y_n , and all unitary constraints over these variables (that is, constraints whose support is one of the variables), and combines all these constraints to get some information (by projecting) over one of the variables, say x .⁶

Such a *local consistency rule* [7] involves a constraint c with support over the set of variables X and a unary constraint c_x with support over the variable $x \in X$, and consists of three phases

- the computation of the solution of the subproblem $\{c_x, c\}$ by computing $c_x \otimes c$,
- the projection of the computed solution over variable x by computing $c'_x = (c_x \otimes c) \Downarrow_x = c_x \otimes (c \Downarrow_x)$, and then
- substituting the original constraint c_x over x with the new one by the assignment $c_x \leftarrow c'_x$.

The application of a local consistency rule leads to an equivalent problem if multiplicative idempotency holds [7]. We relax this condition by performing two assignments at each step of the new rule.

Definition 6 (local consistency rule) A local consistency rule involving a constraint c and a unary constraint c_x with $\text{supp}(c_x) = \{x\} \subset \text{supp}(c)$ consists of the following phases

- substituting the original constraint c_x with c'_x , computed as usual [7]

$$c'_x = c_x \otimes (c \Downarrow_x)$$

- modifying the constraint c in a new constraint c' that takes into account the changes performed on c_x ⁷

$$c' = c \oplus (c \Downarrow_x),$$

where the constraint division function $\oplus : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ is defined as $(c_1 \oplus c_2)\eta = c_1\eta \div c_2\eta$.

⁶ Note that this notion represents a generalized form of arc-consistency, since it was originally defined for binary constraints only [22].

⁷ Since constraint c_x is combined with $c \Downarrow_x$, c' is divided by the same value.

Notice that the two steps of Definition 6 corresponds to the steps performed by the *project* operator used in fair valued CSPs (see e.g. Algorithm 1 in [12, p.207]). The *extend* operator is instead not considered here, but it is worth to notice that it can be emulated by using constraint removal and combination.

The main result of this section is that the application of the above local consistency rules does not change the solution of soft constraint satisfaction problems defined on invertible by residuation semirings.

Proposition 10 (preserving solutions) *Let \mathcal{K} be an absorptive, invertible by residuation semiring, and let us consider a soft constraint satisfaction problem on it. Then, the application of the local consistency rules in Definition 6 does not change the solution of the problem, that is $c_x \otimes c = c'_x \otimes c'$.*

5 Conclusions and further work

Our work represents an investigation on the semiring-based approaches to soft constraints. After revising some basics of (tropical) semirings and of residuation theory, we show how to define a suitable division operator, proving how the latter can be used to generalize current algorithm for local consistency.

The papers that are most related to our study are those by Cooper and Schiex [11, 12, 25]. These authors propose a generalized version of arc-consistency for valued CSPs [5, 26], by defining a difference operation which is the inverse of the aggregation operation (corresponding to our \times operator). As shown in Section 3.4, the two proposals share many similarities. However, our solution differs in two aspects. First of all, it has been explicitly devised for obtaining a total operation that can be applied to absorptive semirings where the induced ordering is partial, such as those arising in set-based CSPs.

More importantly, though, by relying on the notion of residuation, our solution may take advantage on the large family of studies on tropical arithmetics, in particular in finding general criteria establishing when a semiring is invertible by residuation, and for obtaining an algorithmic procedure for the calculus of the result of the division operation (considered as the resolution of a linear equation), as it is illustrated e.g. in [2, Section 4.5]. We consider mandatory a future, exhaustive comparison of our formalism with fair valued CSPs.

Further algorithmic considerations should then be taken into account before proposing an actual implementation of our local consistency rules. For example, note that if the value of a constraint c should coincide with that of its projection $c \downarrow_x$, than the value of the constraint c' in Definition 10 would be 1, thus possibly increasing, hence the chaotic iteration applied to our rules could possibly not terminate. Using a set of *guarded* rules that never perform such (useless) divisions would however guarantee in this case termination.

The papers by Brown, Larrosa, Meseguer, Schiex and Verfaillie consider max and weighted CSPs and define local consistency algorithms also in the presence of a non-idempotent aggregation operator [9, 18, 19, 20, 21]. Our work shares the same aim, even if we do not consider one specific formalism but instead the whole semiring-based approach. It is a matter of further investigation if their proposals could be generalized and applied to our framework.

Current work is now devoted to the introduction of an inverse of the sum operator, and the definition of a framework for soft constraint databases, as well as a throughout analysis of the relationship between these operators in absorptive semirings.

ACKNOWLEDGEMENTS

We are greatly indebted to Nic Wilson for his suggestions, as well as to Paola Cellini, Martin Cooper, Javier Larrosa, and Thomas Schiex for their help in discussing and improving the contents of the paper.

REFERENCES

- [1] L. Aceto, Z. Ésik, and A. Ingólfssdóttir, ‘Equational theories of tropical semirings’, *Theoretical Computer Science*, **298**(3), 417–469, (2003).
- [2] F. Baccelli, G. Cohen, G.J. Olsder, and J-P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*, Wiley, 1992.
- [3] C. Bessière, ‘Arc-consistency and arc-consistency again’, *Artificial Intelligence*, **65**(1), 179–190, (1994).
- [4] S. Bistarelli, *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lect. Notes in Comp. Sci.*, Springer, 2004.
- [5] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie, ‘Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison’, *Constraints*, **4**(3), 199–240, (1999).
- [6] S. Bistarelli, U. Montanari, and F. Rossi, ‘Constraint solving over semirings’, in *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 624–630. Morgan Kaufman, (1995).
- [7] S. Bistarelli, U. Montanari, and F. Rossi, ‘Semiring-based constraint solving and optimization’, *Journal of the ACM*, **44**(2), 201–236, (1997).
- [8] T.S. Blyth and M.F. Janowitz, *Residuation Theory*, Pergamon Press, 1972.
- [9] K. Brown, ‘Soft consistencies for weighted csp’, in *Proc. 5th Int. Workshop on Soft Constraints*, (2003).
- [10] Z. Q. Cao, K. H. Kim, and F. W. Roush, *Incline algebra and applications*, Ellis Horwood, 1984.
- [11] M.C. Cooper, ‘Reduction operations in fuzzy or valued constraint satisfaction’, *Fuzzy Sets and Systems*, **134**(3), 311–342, (2003).
- [12] M.C. Cooper and T. Schiex, ‘Arc consistency for soft constraints’, *Artificial Intelligence*, **154**(1–2), 199–227, (2004).
- [13] S. Gaubert and Max Plus, ‘Methods and applications of (max,+) linear algebra’, in *Proc. 14th Int. Symposium on Theoretical Aspects of Computer Science (STACS)*, eds., R. Reischuk and M. Morvan, volume 1200 of *Lect. Notes in Comp. Sci.*, pp. 261–282. Springer, (1997).
- [14] J. Golan, *Semirings and Affine Equations over Them: Theory and Applications*, Kluwer, 2003.
- [15] M. Gondran and M. Minoux, ‘Linear algebra in dioids: A survey of recent results’, in *Algebraic and Combinatorial Methods in Operations Research*, eds., R.E. Burkard, R.A. Cuninghame-Green, and U. Zimmermann, volume 19 of *Annals of Discrete Mathematics*, 147–164, North-Holland, (1984).
- [16] P. Van Hentenryck, Y. Deville, and C.M. Teng, ‘A generic arc-consistency algorithm and its specializations’, *Artificial Intelligence*, **57**(2–3), 291–321, (1992).
- [17] J. Kuntzmann, *Théorie des Réseaux et Graphes*, Dunod, 1972.
- [18] J. Larrosa, ‘On arc and node consistency in weighted CSP’, in *Proc. 18th National Conf. on Artificial Intelligence, 14th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pp. 48–53. AAAI Press, (2002).
- [19] J. Larrosa, P. Meseguer, and T. Schiex, ‘Maintaining reversible DAC for MAX-CSP’, *Artificial Intelligence*, **107**(1), 149–163, (1999).
- [20] J. Larrosa, P. Meseguer, T. Schiex, and G. Verfaillie, ‘Reversible DAC and other improvements for solving max-csp’, in *Proc. 15th National Conf. on Artificial Intelligence, 10th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pp. 347–352. AAAI Press/The MIT Press, (1998).
- [21] J. Larrosa and T. Schiex, ‘In the quest of the best form of local consistency for weighted csp’, in *Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, eds., G. Gottlob and T. Walsh, pp. 239–244. Morgan Kaufman, (2003).
- [22] A.K. Mackworth, ‘Constraint satisfaction’, in *Encyclopedia of AI*, ed., S. Shapiro, 285–293, Wiley, second edn., (1992).
- [23] J.-E. Pin, ‘Tropical semirings’, in *Idempotency*, ed., J. Gunawardena, 50–69, Cambridge University Press, (1998).
- [24] S. Rudeanu and D. Vaida, ‘Semirings in operations research and computer science’, *Fundamenta Informaticae*, **61**(1), 61–85, (2004).
- [25] T. Schiex, ‘Arc consistency for soft constraints’, in *Proc. 6th Int. Conf. on Principles and Practice of Constraint Programming (CP2000)*, ed., R. Dechter, volume 1894 of *Lect. Notes in Comp. Sci.*, pp. 411–424. Springer, (2000).
- [26] T. Schiex, H. Fargier, and G. Verfaillie, ‘Valued constraint satisfaction problems: Hard and easy problems’, in *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 631–637. Morgan Kaufman, (1995).
- [27] N. Wilson, ‘Bounds and pre-processing for local computation of semiring valuations’, in *Proc. ECAI Workshop on Local Computation for Logics and Uncertainty*, (2004).

Evaluating ASP and commercial solvers on the CSPLib

Marco Cadoli, Toni Mancini, Davide Micaletto and Fabio Patrizi¹

Abstract. This paper deals with three solvers for combinatorial problems: the commercial state-of-the-art solver Ilog OPL, and the research ASP systems DLV and SMODELS. The first goal of this research is to evaluate the relative performance of such systems, using a reproducible and extensible experimental methodology. In particular, we consider a third-party problem library, i.e., the CSPLib, and uniform rules for modelling and selecting instances. The second goal is to analyze the effects of a popular reformulation technique, i.e., symmetry breaking, and the impact of other modelling aspects, like global constraints and auxiliary predicates. Results show that there is not a single solver winning on all problems, and that reformulation is almost always beneficial: symmetry-breaking may be a good choice, but its complexity has to be carefully chosen, by taking into account also the particular solver used. Global constraints often, but not always, help OPL, and the addition of auxiliary predicates is usually worth, especially when dealing with ASP solvers. Moreover, interesting synergies among the various modelling techniques exist.

1 Introduction

The last decade has witnessed a large effort in the development of solvers for combinatorial problems. The traditional approach based on writing *ad hoc* algorithms, complete or incomplete, or translating in a format suitable for Integer Programming solvers², has been challenged by the use of libraries for Constraint Programming (CP), such as Ilog SOLVER³, interfaced through classical programming languages, e.g. C++ or Prolog. At the same time, the need for a higher level of abstraction led to the design and development of 1) General purpose languages for constraint modelling/programming –e.g. OPL [16], XPRESS^{MP}⁴ or GAMS [1]– and 2) Languages based on specific solvers, such as AMPL [7], DLV [8], SMODELS [11] or ASSAT [9].

This paper focuses on the last class of solvers, which are highly declarative, and characterized by the possibility of decoupling the specification of a problem from the instance, and by having optional procedural information. In particular, we consider one commercial state-of-the-art solver, i.e., Ilog OPL and some ASP solvers, namely DLV and SMODELS. The latter has the interesting property of sharing the specification language with several other solvers through the common parser LPARSE⁵. As a matter of fact, such systems exhibit interesting differences, including availability (OPL and ASP are, respectively, payware and freeware systems, the latter often being open source), algorithm used by the solver (resp. backtracking- and fixpoint-based), expressiveness of the modelling language (e.g.,

availability of arrays of finite domain variables vs. boolean matrices), compactness of constraint representation (e.g., availability of global constraints), possibility of specifying a separate search procedure.

The first goal of this research is to evaluate the relative performance of such systems, using a reproducible and extensible experimental methodology. In particular, we consider a third-party problem library, i.e. the CSPLib⁶, and uniform rules for modelling and instance selection. The second goal is to analyze the effects of a popular reformulation technique, i.e. symmetry breaking, and the impact of other modelling aspects, like the use of global constraints and auxiliary predicates.

As for symmetry-breaking, given the high abstraction level of the languages, an immediately usable form of reformulation is through the addition of new constraints (cf., e.g., [3, 6]). Since previous studies [13] showed that this technique is effective when simple formulae are added, it is interesting to know –for each class of solvers– what is the amount of symmetry breaking that can be added to the model, and still improving performances. As a side-effect, we also aim to advance the state of knowledge on the good practices in modelling for some important classes of solvers.

Comparison among different solvers for CP has already been addressed in the literature: in particular, we recall [5] and [17] where SOLVER is compared to other CP languages such as, e.g., OZ [15], CLAIRE⁷, and various Prolog-based systems. Moreover, some benchmark suites have been proposed, cf., e.g., the COCONUT one [14]. Also on the ASP side, which has been the subject of much research in the recent years, benchmark suites have been built in order to facilitate the task of evaluating improvements of their latest implementations, the most well-known being ASPARAGUS⁸ and ASPLib⁹. However, less research has been done in comparing solvers based on different formalisms and technologies, and in evaluating the relative impact of different features and modelling techniques. In particular, very few papers compare ASP solvers to state-of-the-art systems for CP. To this end, we cite [4], where two ASP solvers are compared to a CLP(FD) Prolog library on six problems: Graph coloring, Hamiltonian path, Protein folding, Schur numbers, Blocks world, and Knapsack, and [12], where ASP and Abductive Logic Programming systems, as well as a first-order finite model finder, are compared in terms of modelling languages and relative performances on three problems: Graph coloring, N-queens, and a scheduling problem.

In this research we consider the CSPLib problem library for our experiments. CSPLib is a collection of 45 problems, classified into 7 areas, and is widely known in the CP community. Since many of them are described only in natural language, this work also provides, as a side-effect, formal specifications of such problems in the modelling languages adopted by some solvers.

¹ Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”. E-mail: cadoli|tmancini|micaletto|patrizi@dis.uniroma1.it

² e.g. Ilog CPLEX, cf. <http://www.ilog.com/products/cplex>.

³ cf. <http://www.ilog.com/products/solver>.

⁴ cf. <http://www.dashoptimization.com>.

⁵ cf. <http://www.tcs.hut.fi/Software/smodels>.

⁶ cf. <http://www.csplib.org>.

⁷ cf. <http://claire3.free.fr>.

⁸ cf. <http://asparagus.cs.uni-potsdam.de>.

⁹ cf. <http://dit.unitn.it/~wasp>.

2 Methodology

In this section we present the methodology adopted in order to achieve the two goals mentioned in Section 1. For each problem, we define a number of different formulations, a *base specification*, obtained by a straightforward and intuitive “translation” of the CSPLib problem description into the target language, and several *reformulated* ones, obtained by using different techniques proposed in the literature: (i) symmetry-breaking, (ii) addition of global constraints and (iii) addition of auxiliary predicates. Moreover, in order to establish whether merging different reformulations, which are proven to improve performances when used alone, speeds-up even more the computation, we considered additional specifications for the same problems, obtained by *combining* the aforementioned techniques, and exploring the existence of synergies among them. Finally, a preliminary evaluation of the impact of numbers and arithmetic constraints in all the languages involved in the experimentation has been performed on one problem.

We tried to do the modelling task in a way as systematic as possible, by requiring the specifications of the various solvers to be similar to each other. The criteria followed during the modelling task are discussed in Section 3.2. As for the instances, in this paper we opted for problems which input data is made of few integer parameters (with two exceptions, which are discussed in Section 3.1). In order to have a synthetic qualitative measure of the various solvers performance, for each problem we fix all the input parameters but one. Hence in our results we report for each problem and solver the largest instance (denoted by the value given to the selected parameter) that is solvable in a given time limit (one hour).

3 The experimental framework

3.1 Selecting the problems

So far we have formulated and solved 7 problems, which are listed along with their identification number in CSPLib and the parameter that defines problem instances. Such problems cover the 7 areas of the collection. For space reasons, we omit their descriptions, which can be found at www.csplib.org.

Id	Problem name	Instances defined by
017	Ramsey	# of graph nodes
010	Social golfer (decisional version) (32 players, 8 groups of 4)	Schedule length
006	Golomb rulers	# of marks
001	Car sequencing	Benchmark instances
018	Water buckets	Benchmark instances
032	Maximum density still life	Board size
033	Word design (decisional version)	# of words

Some comments are in order. First of all, we considered the decisional versions of two problems, Social golfer and Word design, since our solvers were unable to achieve the optimal solutions. Secondly, for two problems, Car sequencing and Water buckets, instances could not be naturally encoded by a single parameter. Hence, they have been derived from benchmarks taken from the CSPLib. In particular, as for Car sequencing, we considered benchmarks “4/72”, “6/76” and “10/93”. Unfortunately, they were too hard for our solvers. Hence, from any of them, we generated a set of instances by reducing the number of car classes (cf. the CSPLib problem description) to all possible smaller values, and consequently resizing station capacities in order to avoid an undesirable overconstraining that would make instances unfeasible. Thus, instances derived from the same benchmark could be ordered according to the value for the

(reduced) number of classes, which can be regarded as a measure for their size. As for Water buckets instead, we observe that the CSPLib formulation actually fixes the capacities of the three buckets and both the start and goal states. Hence, we built a specification parametric wrt the start and goal states, and designed 11 instances from that considered in the original problem description, which have been proved to be non-trivial by preliminary experiments. Since such instances could not be denoted by a single parameter, solvers’ performance for this problem and the different specifications have been compared by considering the overall time needed to solve them.

The Water buckets problem has been used also to evaluate the impact of numbers and arithmetic constraints in problem models. In fact, it is well known that such issues may greatly degrade solvers’ performance, and that this behavior strongly depends on the underlying solving algorithm. In order to understand how negatively numbers and arithmetic constraints affect the behavior of the various solvers, we built a new set of instances, obtained by the original one by duplicating the capacities of the buckets (originally 3, 5, and 8), and the amount of water in each of them both in the start and goal states. Of course, the new instances are equivalent to the original ones, having the same set of solutions, but force all solvers to deal with larger domains for numbers.

3.2 Selecting the problem models

As claimed in Section 2, in order to build an extensible experimental framework, we followed the approach of being as uniform and systematic as possible during the modelling phase, by requiring the specifications of the various solvers to be similar to each other. Hence, even if not always identical because of the intrinsic differences among the languages that could not be overcome, all of the models share the same ideas behind the search space and constraints definitions, independently of the language. Below, we discuss the general criteria followed during the modelling phase, and the different formulations considered for each problem¹⁰.

General modelling criteria. The first obvious difference between OPL and the ASP solvers concerns the search space declaration. The former relies on the notion of *function* from a finite domain to a finite domain, while the latter ones have just *relations*, which must be restricted to functions through specific constraints. Domains of relations can be implicit in DLV, since the system infers them by contextual information. For each language, we used the most natural declaration construct, i.e., functions for OPL, and untyped relations for DLV. Secondly, since the domain itself can play a major role in efficiency, sometimes it has been inferred through some *a posteriori* consideration. As an example, in Golomb rulers the maximum position for marks is upper-bounded by 2^m (m being the number of marks), but choosing such a large number can advantage OPL, which has powerful arc-consistency algorithms for domain reduction. As a consequence, we used the upper bound $3L/2$ for all solvers, L being the maximum mark value for the optimum of the specific instance.

Finally, since the performance typically depends on the instances being positive or negative, we considered the *optimization* versions of Ramsey, Golomb rulers, Water buckets, and Maximum density still life problems. As a matter of fact, for proving that a solution is optimal, solvers have to solve both positive and negative instances.

Base specifications. The first formulation considered for each problem is the so called *base specification*. This has been obtained by a straightforward translation of the CSPLib problem description into

¹⁰ cf. <http://www.dis.uniroma1.it/~patrizi/ecai06/encodings.html> for their encodings.

the target language, by taking into account the general criteria discussed above, and, arguably, is the most natural and declarative.

Reformulation by symmetry-breaking. The first kind of reformulation considered aims at evaluating the impact of performing symmetry-breaking. Since we deal with highly declarative languages that exhibit a neat separation of the problem specification from the instances, we adopted the approach of adding further constraints to the base specification that break the structural symmetries of the problems. Symmetry-breaking is dealt with in a systematic way, by considering the general and uniform schemes for symmetry-breaking constraints presented in [10]. These are briefly recalled in what follows (examples below are given in the simple case where all permutations of values are symmetries; generalizations exist):

- **Selective assignment (SA):** A subset of the variables are assigned to precise domain values: a good example is in the Social golfer problem, where, in order to break the permutation symmetries among groups, we can fix the group assignment for the first and partially for the second week.
- **Selective ordering (SO):** Values assigned to a subset of the variables are forced to be ordered: an example is given by the Golomb rulers problem, where, in order to break the symmetry that “reverses” the ruler, we can force the distance between the first two marks to be less than the difference between the last two.
- **Lowest-index ordering (LI):** Linear orders are fixed among domain values and variables, and assignments of variables (x_1, \dots, x_n) are required to be such that, for any pair of values (d, d'), if $d < d'$ then $\min\{i|x_i = d\} < \min\{i|x_i = d'\}$. An example is given by the Ramsey problem: once orders are fixed over colors, e.g. red < green < blue, and over edges, we can force the assignments to be such that the least index of red colored edges is lower than the least index of green colored ones, and analogously for green and blue edges.
- **Size-based ordering (SB):** After fixing a linear order on values, we force assignments to be such that $|\{x \in V|x = d\}| \leq |\{x \in V|x = d'\}|$, for any pair of values $d \leq d'$, V being the set of variables. For example, in the Ramsey problem we could require the number of blue colored edges to be greater than or equal to that of green ones, in turn forcing the latter to be greater than or equal to the number of red colored edges. Generalizations of this schema do exist, depending on the way the partition of the variables set into size-ordered sets is defined.
- **Lexicographic ordering (LX):** This schema is widely applied in case of search spaces defined by matrices, where all permutations of rows (or columns) are symmetries. It consists in forcing the assignments to be such that all rows (resp. columns) are lexicographically ordered.
- **Double-lex (lex^2) ordering (L2):** A generalization of the previous schema, applicable where the matrix has both rows and columns symmetries. It consists in forcing assignments to be such that both rows and columns are lexicographically ordered (cf., e.g., [6]). A good example is Social golfer, in which the search space can be defined as a 2D matrix that assigns a group to every combination player/week. Such a matrix has all rows and columns symmetries (we can swap the schedules of any two players, and the group assignments of any two weeks).

Above schemes for symmetry-breaking can be qualitatively classified in terms of “how much” they reduce the search space (i.e., their *effectiveness*), and in terms of “how complex” is their evaluation. In particular they can be partially ordered as follows: SA < SO < LI < LX < L2, and LI < SB, where $s_1 < s_2$ means that schema s_2

better reduces the search space. However, s_2 typically requires more complex constraints than s_1 .

In many cases, more than a single schema is applicable for breaking the symmetries of a given specification and the problem of choosing what is the “right” amount of symmetry-breaking that is worth adding for a given solver arises. In what follows, we give a partial answer to this question.

Reformulation by adding global constraints. Global constraints (GC) encapsulate, and are logically equivalent to, a set of other constraints. Despite this equivalence, global constraints come with more powerful filtering algorithms, and a specification exhibiting them is likely to be much more efficiently evaluable. One of the most well-known global constraints supported by constraint solvers is `alldifferent(x1, ..., xn)` that forces the labeling algorithm to assign different values to all its input variables. Of course, such a constraint can be replaced by a set of binary inequalities $x_i \neq x_j$ (for all $i \neq j$), but such a substitution will result in poorer propagation, hence in less efficiency. Several global constraints are supported by OPL, e.g., `alldifferent` and `distribute`. According to the problems structure, the former has been applied to Golomb Rulers and the latter to Social golfer, Car sequencing and Word design. As for Ramsey, Water bucket and Maximum density still life none of such reformulations applies.

On the other hand, ASP solvers do not offer such a feature, hence no comparison can be made on this issue.

Reformulation by adding auxiliary predicates. A predicate in the search space is called *auxiliary* if its extensions functionally depend on those of the other ones. The use of auxiliary guessed predicates is very common in declarative programming, especially when the user needs to store partial results, to maintain intermediate states. Although the use of auxiliary predicates increases the size of the search space, in some cases this results in a simplification of complex constraints and in a reduction of the number of their variables, and hence may lead to appreciable time savings.

We consider equivalent specifications, obtained by using auxiliary predicates, for all of the selected problems. However, the bottom-up evaluation algorithms of DLV and SMODELS may significantly advantage ASP solvers over OPL on such specifications, since auxiliary predicates are usually defined in rule-heads. To this end, when adding auxiliary predicates to OPL specifications, we also added simple search procedures instructing the labelling algorithm to not branch on auxiliary variables, while maintaining the default behavior on the others.

Reformulation by combining different techniques. In many cases, more than one single reformulation strategy improves performances on a given problem. Hence, the question arises whether synergies exist among them, and what techniques are likely to work well together, for each solver. To this end, for each problem we consider two additional formulations: the first one has been obtained, according to the aforementioned uniformity criteria, by merging the two reformulations (among symmetry-breaking, addition of global constraints and of auxiliary predicates) that, for each solver, resulted to be the most performant. Finally, in order to understand whether there exist better, undiscovered synergies, we relaxed the uniformity hypothesis, and considered some of the other possible combinations of reformulation strategies, with the goal to further boost performances.

4 Experimental results

Our experiments have been performed by using the following solvers: *i*) Ilog SOLVER v. 5.3, invoked through OPLSTUDIO 3.61,

ii) SMODELS v. 2.28, by using LPARSE 1.0.17 for grounding, iii) DLV v. 2005-02-23, on a 2 CPU Intel Xeon 2.4 Ghz computer, with a 2.5 GB RAM and Linux v. 2.4.18-64GB-SMP.

For every problem, we wrote the specifications described in Sections 2 and 3 in the different languages. We then ran the different specifications for each solver on the same set of instances, with a timeout of 1 hour. Table 1 shows a summary of the results concerning base specifications and their various reformulations. In particular, for each problem and solver, we report the largest instance the various systems were able to solve (in the given time-limit) for the base specifications, and the improvements obtained by the different reformulations. A 0^+ (resp. 0^-) means that the size of the largest instance solved within the time limit was unchanged, but absolute solving times were significantly lower (resp. higher). As for Car sequencing, we report, for each set of instances generated from CSPLib benchmarks “4/72”, “6/76” and “10/93”, the largest one solved, i.e., the one with the largest number of classes (cf. the discussion about instance selection for this problems in Section 3.1), and, as for Water buckets, the overall time needed to solve the whole set of instances (instances that could not be solved contributed with 3,600 seconds to the overall time).

Impact of symmetry-breaking. From the experiments, it can be observed that symmetry-breaking may be beneficial, although the complexity of the adopted symmetry-breaking constraints needs to be carefully chosen. As an example, DLV performs much better on the Ramsey problem with LI symmetry-breaking constraints, but it is slowed down when the more complex SB schema is adopted. A similar behavior can be observed on SMODELS.

As for Social golfer, Table 1 does not show significant performance improvements when symmetry-breaking is applied, with the ASP solvers (especially SMODELS) being significantly slowed down when adopting the most complex schemas (LX and L2). However, it is worth noting that, on smaller negative (non-benchmark) instances, impressive speed-ups have been obtained for all systems, especially when using SA. As for LX, we also observe that it can be applied in two different ways, i.e., forcing either players’ scheduling or weekly groupings to be lexicographically ordered. Values reported in Table 1 are obtained by lexicographically ordering weekly groupings: as a matter of fact, ordering players’ scheduling is even less performant on SMODELS, being comparable for the other solvers. General rules for determining the right “amount” of symmetry breaking for any given solver on different problems are currently still unknown, but it seems that the simplest ones, e.g., SA, have to be preferred when using ASP solvers.

Impact of global constraints. Experiments confirm that OPL may benefit from the use of global constraints. As an example, the base specification of the Golomb rulers problem encodes the constraint that forces the differences between pairs of marks to be all different by a set of binary inequalities. By replacing them with an `alldifferent` constraint, OPL was able to solve the instance with 11 marks in the time-limit, and time required to solve smaller instances significantly decreases. Also the Social golfer specification can be restated by using global constraints, in particular the `distribute` constraint. However, in this case our results show that OPL does not benefit from such a reformulation, in that it was not able to solve even the 4-weeks instance (solved in about 11 seconds with the base specification). Global constraints help OPL also on other problems, i.e., Car sequencing, where `distribute` can be used, even if the performance improvements don’t make it able to solve larger instances. Finally, Word Design seems not to be affected by the introduction of `distribute`.

Impact of auxiliary predicates. ASP solvers seem to benefit from the use of auxiliary predicates (often not really needed by OPL, which allows to express more elaborate constraints), especially when they are defined relying on the minimal model semantics of ASP (hence, in rule heads). As an example, SMODELS solves the 6-weeks instance of the Social golfer problem in 6 seconds, when the auxiliary `meet/3` predicate is used, while solving the base specification requires 41 minutes. Even if not as much remarkable, a similar behavior is observed in DLV. Similar results have been obtained for Ramsey, where the auxiliary predicate `color_used/1` is added.

Using the `meet` auxiliary predicate helps also OPL (but only after a simple search procedure that excludes branches on its variables has been added, cf. Section 3.2). In particular, the 5-weeks instance has been solved in just 8 seconds (with respect to the 80 seconds of the base specification). It is interesting to note that, by adding a smarter search procedure, solving time dropped down to less than a second. This is a good evidence that exploiting the peculiarities of the solver at hand may significantly increase the performance.

Synergic reformulations. Specifications obtained by combining, for each problem and solver, the most two performant techniques, in many cases further boost performances, or at least do not affect them negatively. This is the case, e.g., of Social Golfer, Golomb Rulers, and Word design problems when solved by OPL, that proved to be able to deal with larger instances, and Car sequencing, where solving times significantly decreased. Few exceptions do exist, i.e., Ramsey and Maximum density still life, where solving times were a bit higher, but did not prevent OPL to deal with the largest instances previously solved. Similar results can be observed for SMODELS and DLV. This gives evidence that combining “good” reformulations is in general a promising strategy to further boost performance of all solvers. Table 1 also shows some of the results obtained by other possible combinations, without considering any uniformity criteria. It can be observed that in few cases even better results could be achieved (cf., e.g., OPL on Social golfer and the specification with auxiliary predicates and global constraints), but in several others, only worse performances were obtained.

Impact of numbers and arithmetic constraints. Experiments performed on Water buckets, with the most performant specification (i.e., that with auxiliary predicates) on instances obtained by doubling both the buckets capacities and the water contents in the start and goal states, confirm that numbers and arithmetic constraints are a major obstacle for all solvers. In particular, the time needed by OPL to solve the whole set of instances is almost 13 times higher, while for DLV and SMODELS the solving time increases, respectively, of about 130 and 45 times.

5 Conclusions

In this paper we reported results about an experimental investigation which aims at comparing the relative efficiency of a commercial backtracking-based and two academic ASP solvers. In particular, we modelled a number of problems from the CSPLib into the languages used by the different solvers, in a way as systematic as possible, by also applying symmetry-breaking, using global constraints and auxiliary predicates, and evaluating synergies among such techniques. Results show that there is not a single solver winning on all problems, with ASP being comparable to OPL for many of them, and that reformulating the specification almost always improves performances. However, even if our experiments suggest some good modelling practices, an exact understanding of which reformulations lead to the best performances for a given problem and solver remains a

Problems	OPL				DLV			SMODELS				
Ramsey (# of nodes)	Base 16	LI -3	SB 0-	Aux 0	Base 9	LI +5	SB -1	Aux 0+	Base/Aux*	LI 9 +7	SB -1	
	Aux-SB 0-	Aux-LI 0-		Aux-LI +7	Aux-SB -1							
Social Golfer (# of weeks)	Base 5	SA 0	L2 0	LX 0	Base 6	SA 0	L2 -2	LX 0	Base 6	SA 0+ -6	L2 -6	LX -6
	Aux 0+	GC -2	Aux-SA 0+	Aux-GC +1	Aux 0+	Aux-SA 0+	Aux-LX 0		Aux 0+	Aux+SA 0+	Aux+LX -6	
Golomb Rulers (# of marks)	Base 10	SO 0+	Aux +1	GC +1	Base 9	SO 0	Aux 0	Base 6	SO 0	Aux +2		
	Aux-GC +1	Aux-SO-GC +2		Aux-SO 0				Aux-SO +2				
Car Sequencing bench. 4/72 (# of classes)	Base 10	SO 0	Aux 0+		Base 13	SO 0	Aux 0	Base 13	SO 0	Aux 0		
	Aux-GC 0+	Aux-SO-GC 0+		Aux-SO 0+				Aux-SO 0+				
Car Sequencing bench. 6/76 (# of classes)	Base 6	SO 0	Aux 0		Base 9	SO 0	Aux 0	Base 9	SO 0	Aux 0		
	Aux-GC 0	Aux-SO-GC 0		Aux-SO 0+				Aux-SO 0+				
Car Sequencing bench. 10/93 (# of classes)	Base 12	SO 0	Aux 0		Base 12	SO 0	Aux 0	Base 12	SO 0	Aux 0		
	Aux-GC 0	Aux-SO-GC 0		Aux-SO 0+				Aux-SO 0+				
Water Bucket (total time)	Base 54.77 s	Aux 18.22 s			Base 1332.62 s	Aux 202.00 s		Base 9765.8 s	Aux 173.43 s			
Maximum Density Still Life (board size)	Base 8	SO 0	SB 0	Aux 0-	Base 7	SO 0	SB 0	Aux -1	Base 7	SO +1	SB 0	Aux +1
	Aux-SO 0-	Aux-SB 0-		Aux-SO 0	Aux-SB 0		Aux-SO +1	Aux-SB 0				
Word Design DNA (# of words)	Base 86	SO 0	SB -30	LX +1	Base 5	SO 0	SB 0	LX 0	Base 11	SO 0+	SB -1	LX +3
	Aux 0	GC 0	Aux-LX +1	Aux-GC 0	Aux 0	Aux-SO 0	Aux-LX 0		Aux +10	Aux-LX +1	Aux-SO +35	

* Since for Ramsey problem the use of auxiliary predicates seems to be unavoidable in SMODELS, the Base and Aux specifications coincide.

Table 1. Sizes of the largest instances solved by OPL, DLV, and SMODELS in 1 hour, using the base specification and their reformulations. Variations are given wrt the base specifications. Bold numbers denote the best results.

challenge. Finally, we also started an investigation about the impact of numbers and arithmetic constraints in problem specifications.

Our current efforts are aimed at covering a larger part of the CSPLib, extending the set of reformulations involved (e.g., using implied constraints) and considering other systems (e.g. ASP or SAT based).

6 Acknowledgements

The authors would like to thank the anonymous reviewers, in particular for suggesting a better Water Bucket SMODELS encoding.

REFERENCES

- [1] E. Castillo, A. J. Conejo, P. Pedregal, R. Garca, and N. Alguacil, *Building and Solving Mathematical Programming Models in Engineering and Science*, John Wiley & Sons, 2001.
- [2] P. Codognet and D. Diaz, ‘Compiling constraints in clp(FD)’, *J. of Logic Programming*, **27**, 185–226, (1996).
- [3] J. M. Crawford, M. L. Ginsberg, E. M. Luks, and A. Roy, ‘Symmetry-breaking predicates for search problems’, in *Proc. of KR’96*, pp. 148–159, Cambridge, MA, USA, (1996). Morgan Kaufmann, Los Altos.
- [4] A. Dovier, A. Formisano, and E. Pontelli, ‘A comparison of CLP(FD) and ASP solutions to NP-complete problems’, in *Proc. of ICLP 2005*, volume 3668 of *LNCS*, pp. 67–82, Sitges, Spain, (2005). Springer.
- [5] A. J. Fernández and P. M. Hill, ‘A comparative study of eight constraint programming languages over the Boolean and Finite Domains’, *Constraints*, **5**(3), 275–301, (2000).
- [6] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh, ‘Breaking row and column symmetries in matrix models’, in *Proc. of CP 2002*, volume 2470 of *LNCS*, p. 462 ff., Ithaca, NY, USA, (2002). Springer.
- [7] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Intl. Thomson Publ., 1993.
- [8] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello, ‘The DLV System for Knowledge Representation and Reasoning’, *ACM Trans. on Comp. Logic*. To appear.
- [9] F. Lin and Z. Yuting, ‘ASSAT: Computing answer sets of a logic program by SAT solvers’, *Artif. Intell.*, **157**(1–2), 115–137, (2004).
- [10] T. Mancini and M. Cadoli, ‘Detecting and breaking symmetries by reasoning on problem specifications’, in *Proc. of SARA 2005*, volume 3607 of *LNAI*, pp. 165–181, Airth Castle, Scotland, UK, (2005). Springer.
- [11] I. Niemelä, ‘Logic programs with stable model semantics as a constraint programming paradigm’, *Annals of Math. and Artif. Intell.*, **25**(3,4), 241–273, (1999).
- [12] N. Pelov, E. De Mot, and M. Denecker, ‘Logic Programming approaches for representing and solving Constraint Satisfaction Problems: A comparison’, in *Proc. of LPAR 2000*, volume 1955 of *LNCS*, pp. 225–239, Reunion Island, FR, (2000). Springer.
- [13] A. Ramani, F. A. Aloul, I. L. Markov, and K. A. Sakallah, ‘Breaking instance-independent symmetries in exact graph coloring’, in *Proc. of DATE 2004*, pp. 324–331, Paris, France, (2004). IEEE Comp. Society Press.
- [14] O. Shcherbina, A. Neumaier, D. Sam-Haroud, X.-H. Vu, and T.-V. Nguyen, ‘Benchmarking global optimization and constraint satisfaction codes’, in *Proc. of COCOS 2002*, volume 2861 of *LNCS*, pp. 211–222, Valbonne-Sophia Antipolis, France, (2003). Springer.
- [15] G. Smolka, ‘The Oz programming model’, in *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, 324–343, Springer, (1995).
- [16] P. Van Hentenryck, *The OPL Optimization Programming Language*, The MIT Press, 1999.
- [17] M. Wallace, J. Schimpf, K. Shen, and W. Harvey, ‘On benchmarking constraint logic programming platforms. response to [5]’, *Constraints*, **9**(1), 5–34, (2004).

Automatic Generation of Implied Constraints

John Charnley, Simon Colton¹ and Ian Miguel²

Abstract. A well-known difficulty with solving Constraint Satisfaction Problems (CSPs) is that, while one formulation of a CSP may enable a solver to solve it quickly, a different formulation may take prohibitively long to solve. We demonstrate a system for automatically reformulating CSP solver models by combining the capabilities of machine learning and automated theorem proving with CSP systems. Our system is given a basic CSP formulation and outputs a set of reformulations, each of which includes additional constraints. The additional constraints are generated through a machine learning process and are proven to follow from the basic formulation by a theorem prover. Experimenting with benchmark problem classes from finite algebras, we show how the time invested in reformulation is often recovered many times over when searching for solutions to more difficult problems from the problem class.

1 Introduction

Constraint programming is a successful technology for tackling a wide variety of combinatorial problems in disparate fields, such as scheduling, industrial design, and combinatorial mathematics [19]. To use constraint technology to solve a constraint satisfaction problem (CSP), its solutions must first be characterised, or *by a set of constraints on a set of decision variables. Constraints provide a rich language, so typically many alternative models exist for a given problem, some of which are more effective than others. Formulating an effective constraint program is a challenging task, for which new users lack expertise. Great value is placed, therefore, on any method that helps to reduce this modelling bottleneck.*

One method of improving a model is to introduce constraints implied by the model. We describe here a method for introducing such *implied constraints*. As described in §1.1, this fully automated method is a significant improvement upon the semi-automated method presented in [7]. Given a set of problems from the same problem class, as detailed in §2, we employ a machine learning module to induce relationships in the solutions for small problems and we use a theorem prover to show that the truth of the relationships follow from the CSP model. As described in §3, the relationships are translated and interpreted as implied constraints which are added to the CSP model. On larger instances of the problem class, the additional constraints often lead to a much reduced solving time. In §4, we present the results of applying this approach to some benchmark CSPs, namely finite algebras. We show how the system was able to improve upon the basic solver model in a number of cases and the reformulation time was soon recovered when searching for larger examples. For instance, in one case, we were able to reduce the solving time to 16% of the original, saving around 23 hours of processing.

1.1 Background

In [7], Colton and Miguel describe the application of machine learning and theorem proving to the generation of implied constraints. They employed descriptive machine learning to generate hypotheses about concepts expressed in the basic model of a class of CSPs. They then used a theorem prover to show that certain hypotheses followed from the CSP model, and hence they could be added as implied constraints. While they achieved significant increases in efficiency in this manner, their approach was semi-automatic. In particular, both Miguel and Colton - with expertise in constraint modelling and the domain of application (pure mathematics) respectively - were required to interpret the output from the learning system as constraints and translate them into the language of the constraint solver. Consequently, although the reformulations provided an improvement in solver efficiency, this was offset by the large amount of time and expertise required during the reformulation process.

As described in the next section, our approach makes use of a number of AI techniques to fully automate the process that Colton and Miguel introduced. For brevity, we assume general familiarity with automated theorem proving [8], constraint solving [15] and machine learning [17]. We combine techniques from these areas via usage of the following systems:

- The Otter automated theorem prover, which uses the resolution method to prove theorems by refutation [16].
- The Constraint Logic Programming in Finite Domains (CLPFD) library [4] included with SICStus Prolog.
- The HR automated theory formation system. This performs descriptive learning to speculatively invent concepts and form hypotheses in a domain of interest. HR is extensively described in [6].

Our approach is distinct from, and complementary to, existing automated constraint reformulation methods. For example, the CGRASS system [10] captures common patterns in the hand-transformation of constraint satisfaction problems in transformation rules. Given an input model, CGRASS applies the rules in a forward chaining manner to improve the model gradually. A key difference from the approach presented here is that CGRASS is applicable to single problem instances only, rather than problem classes. The O'CASEY system [13, 14] employs case-based reasoning to capture constraint modelling expertise. Each case records a problem/model pair. Given a new problem, the most similar problem is retrieved from the case base. A model for the new problem is then constructed by following the modelling steps used in the retrieved case. Bessiere et al. present a method for learning the parameters of certain types of implied constraint by considering solutions and non-solutions of relaxations of the original problem [3]. The CONACQ system [1, 2] uses a version space approach to learn a CSP model from scratch given only positive and negative example solutions to the CSP.

¹ Department of Computing, Imperial College, London, United Kingdom

² School of Computer Science, University of St. Andrews, United Kingdom
email: jwc04@doc.ic.ac.uk, sgc@doc.ic.ac.uk, iamm@dcs.st-and.ac.uk

2 The Reformulation Process

A CSP *problem class* is a possibly infinite set, $\{P_1, P_2, \dots\}$, of constraint satisfaction problems. *Instances* of the class are obtained by giving values for the *parameters* of the class. We focus on classes parameterised by a single positive integer n . The number of variables in P_n increases with n , making successive members of the class increasingly difficult to solve. However, the core constraints for each P_i can be expressed using the same general relationships over the variables. We have been working with CSPs where each variable has the same domain. For problem instance P_n from a class of CSPs, this domain is $\{1, \dots, n\}$. Hence, in using the term *domain size*, we mean the size of the domain of any variable, namely n for P_n . A good example of such a class of CSPs is Latin squares, where P_n represents the problem of filling in an n by n grid with the integers 1 to n such that no number appears twice in any row or column.

Our aim is to take the core model of a problem class and automatically produce a reformulated core model containing additional constraints. The additional constraints will be proven to be implied by the core model, so that a solution to any P_n is also a solution to the reformulated version of P_n , and vice versa. We have automated the following approach to reformulating CSPs:

1. The user supplies information about a problem class, including the core constraints. We call this formulation of the CSP the *basic model*.
2. The CLPFD solver generates solutions to small problem instances.
3. The solutions are given, along with the core model, to the HR system. This generates empirically true conjectures which relate concepts expressed in the core constraints.
4. Each conjecture is passed to Otter, and an attempt is made to show that the conjecture is implied by the core model.
5. Each proved conjecture is interpreted as an implied constraint and translated into the syntax of the CLPFD solver. As described in §3, automating this aspect of the process was the most challenging.
6. Each implied constraint is added to the basic model to produce a reformulated model. The small problem instances are then solved again using the reformulated model. Every implied constraint used in a reformulation which improves – in terms of efficiency – upon the basic model, is recorded in a set E .
7. Every pair of constraints from E are added to the basic model and tested for an efficiency increase. Every triple of constraints is then tested, etc., until a user-defined limit is reached, or no reformulation improves upon the best model so far in a particular round.
8. All reformulations leading to a speed up for the small problem instances are presented to the user in decreasing order of the efficiency gained from the reformulation.

As an example, we started with the class of QG3 quasigroup problems (defined in §4). We formed the basic model for this from the first order axioms of QG3 quasigroups, using the translator in our system. Using Sicstus to solve small problems from this class, our system generated 8 QG3 examples. HR used these to discover 100 conjectures about QG3 quasigroups, all of which were proved by Otter. The first round of testing highlighted: (i) $\forall b c d ((b * b = c \wedge d * d = c) \rightarrow (b = d))$ and (ii) $\forall b c d ((b * c = d \wedge c * b = d) \rightarrow (b = c))$ as theorems which – when interpreted as constraints – improve the efficiency of the basic model. The constraints arising from these theorems were further combined into a single model in the second round. In total, the reformulation process took approximately 8 minutes, and produced a much improved model (see §4).

3 Constraint Interpretation

The three AI systems we have been working with are all able to work with first order representations. We note that CSP is in NP, and, from Fagin's theorem [9], NP is the set of languages expressible via existential second-order logic. However, in order to provide an initial benefit via commonality of language, at present, we have limited ourselves to working with CSPs that can be expressed in first order logic.

The HR system has been developed to work with Otter-style first order representations, so the major difficulty of translation was in terms of converting conjectures from Otter syntax to CLPFD constraints. Given a proven conjecture from HR expressed as a string in Otter-style first order syntax, the translation process first tokenizes the string and then parses it using a bespoke Definite Clause Grammar (DCG). For example, the first-order string:

$$\text{all } a b (\exists c d (a * c = b \wedge d * a = b))$$

would be parsed as the following expression:

$$\begin{aligned} \text{all}([\text{var}(a), \text{var}(b)], \exists &[\text{var}(c), \text{var}(d)], \\ e(\&, e(=, e(*, \text{var}(a), \text{var}(c)), \text{var}(b)), \\ e(=, e(*, \text{var}(d), \text{var}(a)), \text{var}(b)))) \end{aligned}$$

Once parsed, expressions are translated into constraints in CLPFD syntax. This is achieved via recursion through the nested expression which creates, in most instances, a new predicate for each level of nesting. How the interpretation method deals with quantifiers, expressions and solution variables is described in §3.1 and §3.2 below. To improve the computational efficiency of the implied constraints, further processing via simplification and further interpretation via contraposition and case splitting is undertaken, as described in §3.3.

3.1 Quantifiers

The DCG produces partially translated universal quantifiers in the form $\text{all}(\text{variable_list}, \text{sub_expression})$. This states that sub_expression should hold for each combination of the values of the variables in variable_list . This is further translated by introducing three new predicates. The first predicate creates a list of all possible value combinations (pairs, triples, etc.) as a list of sets determined by the domain size. The second predicate recurses through this list, and makes a call to the third predicate for each value combination. The third predicate represents sub_expression and is constructed accordingly. The next recursion level is passed details of how the first order variables have so far been translated into Prolog variables, so they can be correctly matched.

As an example, the translation of:

$$\text{all}([\text{var}(a), \text{var}(b)], e(=, \text{var}(a), \text{var}(b)))$$

produces the following triple of logic programs:

$$\begin{array}{lll} \text{A_1(N):-} & \text{A_2([]).} & \text{A_3(V1,V2):-} \\ \text{sub_sets(N,2,S),} & \text{A_2([[V1,V2]|Ss]):-} & \text{V1 \#= V2.} \\ \text{A_2(S).} & \text{A_3(V1,V2),} & \\ & \text{A_2(Ss).} & \end{array}$$

Here, N is the domain size and S is a list of value combinations (in this case, pairs) for $\text{var}(a)$ and $\text{var}(b)$, which is produced by the pre-defined sub_sets predicate. Predicate A_1 creates this list and passes it to A_2 , which recursively calls A_3 for each combination.

Existential quantifiers are partially translated to the form $\exists(\text{variable_list}, \text{sub_expression})$, which states that

sub_expression holds for at least one combination of the values of the variables in *variable_list*. We use two methods to translate existential quantifiers. One method creates all combinations of variable values as before and, using the solver's disjunction syntax, produces another predicate which constrains that at least one case should hold. The second method, which is more effective where it can be used, involves representing the existentially quantified variables as new solution variables in the CSP. These new variables are given domains appropriate to the size of the problem and are added to the list of required solution variables. Of course, in finding a solution, the solver must ensure that these variables are given a value, which enforces their existence.

As an example, using the second method with the following partially translated sentence:

```
exists([var(a), var(b)], e(=, var(a), var(b)))
```

produces the following pair of predicates:

B_1(N,OldVarList,NewVarList):- domain(V1,1,N), domain(V2,1,N), B_2(V1,V2), append(OldVarList,[V1,V2],NewVarList).	B_2(V1,V2):- V1 #= V2.
---	---------------------------

Here, predicate B_1 creates two new solution variables, V1 and V2, to enforce the existence of valid values for first order variables *a* and *b*. These values are passed to B_2, which constrains that their value be equal for whatever instantiation the solver chooses to give them.

3.2 Expressions and Solution Variables

Expressions are partially translated as: $e(op, _, _)$, for instance $e(\&, sub_exp_a, sub_exp_b)$ and $e(=, sub_exp_a, sub_exp_b)$. Depending upon the domain of investigation, particular expressions represent solution variables. For example, in single-operator finite algebras, the solution variables are the result of the algebraic operator on all pairs of elements. Consequently, any expression in the form $e(*, n, m)$, where *n* and *m* are ground integers, identifies a specific solution variable. In other domains, different expressions represent the solution variables, for instance the inverse operator in groups.

The majority of first-order expressions that HR produces can be translated such that individual solution variables are explicitly identified. In some instances, however, where operators are nested, it is not possible to identify the exact solution variables that are affected, as they may vary according to the parameters of the constraint. For example, consider the sentence $\forall a b ((a * b) * (b * a) = b)$. Here, we must first know the results of $a * b$ and $b * a$ before we can identify the solution variable whose value must be constrained equal to *b*. The translator identifies nesting and employs the CLPFD *element* constraint, which is common to many solver systems. This built-in constraint is optimised to resolve values within lists during search.

In general, a conjunction is translated using standard Prolog comma syntax. Equality expressions equate the value of two evaluable sub-expressions. For example, $\forall a b (a * b = b * a)$ equates the result of $a * b$ with that of $b * a$. When translating equalities, a new variable is generated to represent this equated result and each sub-expression is recursively translated in such a way that its result is this new variable. For example, $\forall a b (a * b = b * a)$ generates the following triple of predicates:

C_1(N):- sub_sets(N,2,S), C_2(S).	C_2([],_) :- C_2([V1,V2] Ss):- C_3(V1,V2), C_2(Ss).	C_3(V1,V2):- mv(V1,V2,V3), mv(V2,V1,V3).
---	---	--

Here, C_1 and C_2 handle universal quantification. Predicate C_3 calls the mv(*x,y,z*) predicate which finds the solution variable representing $x * y$ and constrains it to be equal to *z*. Equality of the sub-expressions, namely $a * b$ and $b * a$, is enforced by having both calls to mv return V3.

CLPFD syntax requires some expressions, such as implications and negations, to be expressed using reification variables to represent the constraints for the sub-expressions. Reification variables, which take the value *true* if the constraint holds and *false* otherwise, allow the solver to efficiently propagate constraints during search and are attached to constraints using the CLPFD $\#<=>$ syntax. The sub-expressions of expressions requiring reification must also be reified. Consequently, the translator is able to translate all expressions both with or without reification as necessary.

3.3 Case-splitting and Simplification

The translation process can identify where conjectures can be interpreted as case-splits over quantified variable values. Case-splits allow us to consider specific sets of solution variables and can reduce the complexity of some conjectures by expressing them as simple and efficient constraints. Consider, for example, the theorem: $\forall a b ((a * b = b * a) \rightarrow (a = b))$, which states an anti-Abelian property (that no two distinct elements commute). The contra-position of this theorem is: $\forall a b ((a \neq b) \rightarrow (a * b \neq b * a))$, which can be interpreted as a case-split on the variables *a* and *b*, i.e., when they are equal and when they are not equal. Only when they are unequal do we post the inequality constraint.

The translator finds case-splits by considering whether or not the sub-expressions of an implication contain solution variables. If they do not, then the implication is translated as a case-split using the theorem or its contra-position as appropriate. Case-split translations introduce an additional predicate which succeeds if the variable values meet the case-split requirements, in which case the constraints are posted, and fails otherwise. As an example, the translation of the above anti-Abelian property, after universal quantification, is shown below:

D_1(V1,V2):- D_2(V1,V2), D_3(V1,V2).	D_2(V1,V2):- V1 \= V2.	D_3(V1,V2):- mv(V1,V2,V3), mv(V2,V1,V4), V3 \= V4.
--	---------------------------	---

In this example, D_1 represents $((a \neq b) \rightarrow (a * b \neq b * a))$. The predicate D_2 is used to determine the case, if *a* and *b* (V1 and V2) are unequal then it will succeed and a call to D_3 will be made, posting the constraint $(a * b \neq b * a)$. If it fails, i.e., $a = b$, then no constraints are posted.

In some cases, it is possible to remove variables or re-translate theorems without affecting meaning. In some cases, such simplification can improve the effectiveness of the constraints produced from the theorem. For example, $\forall a b c ((a * b = c \wedge b * a = c) \rightarrow (a = b))$ can be simplified to $\forall a b ((a * b = b * a) \rightarrow (a = b))$. Here we see that the variable *c* – which is simply a marker for equality – has been removed. Care is taken to avoid inappropriate simplification. For example, in the theorem: $\forall a b c ((b * c = c \wedge c * d = c) \rightarrow (b = d))$, we cannot simply remove the variable *c* without changing the semantics of the theorem.

Algebra	No. ex.	Test order	HR time	Test time	Total time	Proportion (%)
QG7	12	6	9:31	6:29	16:00	8.6
QG6	2	6	11:35	1:37	13:12	12.2
Group	22	5	3:31	21:00	24:32	63.1
QG3	8	5	5:32	1:55	7:28	68.8
QG4	20	5	4:29	2:28	6:56	70.4
Ring	25	5	9:17	19:48	29:06	70.6
QG5	8	5	12:57	4:15	17:13	71.7
Moufang	25	3	8:07	7:37	15:44	85.8
QG1	16	4	15:26	7:01	22:28	100.0
QG2	16	4	20:48	6:51	27:39	100.0
Loop	16	5	6:42	3:07	9:50	100.0
Medial	22	3	24:08	2:37	26:45	100.0
Semigroup	23	3	4:15	1:39	5:55	100.0
Monoid	32	4	4:32	4:55	9:28	100.0

Table 1. Reformulation times (minutes:seconds) and proportion of basic model solving time for training (small) domain sizes

4 Experiments and Results

Finite algebras provide first-order benchmark tests for CSP solvers. In particular, QG-quasigroup existence problems have driven much research in constraint solving and other computational techniques. QG-quasigroups are Latin squares with extra axioms, e.g., QG3 quasigroups are Latin squares for which $\forall a b ((a * b) * (b * a) = a)$. Further details are available at the CSPLib.³

Our experimental setup was as follows. We started with the axioms of a particular finite algebra, which effectively defines the core model of a CSP problem class. The axioms were supplied in Otter-style first order syntax and were translated into the basic model of a CSP using the translation process described above. The system ran the basic solver model for sizes up to n , and stopped when the time to solve for size $n + 1$ took over 2 minutes. In this context, and for all the experiments, by *solve* we mean exhausting the search space and finding all solutions. The solutions for the small sizes were passed to HR, which ran until it found 100 proven conjectures (with Otter providing the proofs).

The setup for HR was similar to that in [7]. However, we also supplied two additional background concepts, namely: the concept of two elements in the algebra being equal and two elements being unequal. We have found that this increases the yield of computationally effective constraints found by HR. In addition, we set up HR so that it used Otter to discard any conjectures which were tautologies, i.e., could be proved without the axioms of the domain. This slightly increased HR's processing time, but greatly reduced the number of reformulations to test, hence improved overall efficiency.

The proved conjectures from HR were interpreted and translated as constraints then used to reformulate the CSP using the process described above. For any reformulation shown to improve efficiency on the small – training – problem instances, we compared performance against that of the basic model for problems of size $n + 1$ and larger. Table 1 shows the number and size of small examples found during the reformulation stage, the time taken by HR, the time taken to construct good reformulations and the total reformulation time. This table is ordered by decreasing efficiency gain. Where we improved upon the basic model, table 2 shows the results of taking the best reformulation and applying it to larger problem sizes.

In 8 out of 14 cases, a gain in efficiency was achieved via the reformulated model. Unfortunately, there is a memory bound of 256 Mb. in the Sicstus solver, which effectively meant that we couldn't

Algebra	Domain size	Basic model	Reformulated model	Proportion (%)
QG3	7	3:09	1:24	44.5
	8	10:07:02	3:10:03	31.3
QG4	6	0:07	0:04	54.2
	7	11:19	5:18	46.8
QG5	7	1:24	1:17	91.7
	8	38:05	28:52	75.8
QG6	9	27:25	6:25	23.4
	10	24:21:00	5:53:03	24.2
QG7	8	19:12	3:33	18.5
	9	27:12:35	4:19:42	15.9
Group	8	16:37	4:15	25.6
	9	4:36:39	28:27	10.3
Moufang	4	0:11	0:08	72.3
	5	10:49	4:19	39.9
Ring	7	0:37	0:30	79.8
	8	4:22	2:09	49.5

Table 2. Solving times (hours:minutes:seconds) and proportion of basic model solving time for testing (larger) domain sizes. Note that only those algebras where a speed-up was identified in table 1 are shown

perform tests for problem instances larger than those shown in table 2. In those cases where we did manage to run longer tests, the benefit of reformulation is clear. In particular, for QG3, QG6, QG7 quasigroups and groups, the time taken to produce the reformulated model was a fraction of the time gained in running the larger tests. In the other cases, it seems likely that the reformulation time would be similarly gained back at larger domain sizes.

In [7], Colton and Miguel studied reformulations for QG-quasigroup problems, which were generated in a partially automated, partially hand-crafted way. It is interesting to note that the fully automated method described here re-created all the best reformulations found by hand in [7], with two exceptions. The first exception was with the reformulations of QG3 and QG4 quasigroups. Here, the best fully automated reformulation for both QG3 and QG4 uses the constraints generated from this pair of theorems:

$$\begin{aligned} \forall b c d ((b * c = d \wedge c * b = d) \rightarrow (b = c)) \\ \forall b c d ((b * b = d \wedge c * c = d) \rightarrow (b = c)) \end{aligned}$$

In [7], however, their best reformulations also used another theorem: $\forall a b (a * a = b \rightarrow b * b = a)$. In our experiments, this theorem was found in some reformulations for QG3 and QG4 which improved efficiency. However, these reformulations were less efficient than the best reformulation. The two theorems above state, respectively, that these quasigroups are anti-Abelian (i.e., no two distinct elements commute) and that the main diagonal elements must all be distinct. As described in §3.3, the system uses simplification and the contra-positive of each of these to post constraints for all cases where $b \neq c$. To illustrate these constraints, at size 6 the first of these theorems results in 15 inequality constraints of the form $X_{ij} \neq X_{ji}$ for $1 \leq i, j \leq 6$ and $i \neq j$.

The second exception represents one of numerous instances where the system discovered implied constraints which were not identified in [7]. In particular, the best reformulation for QG5 quasigroups from [7] was identified as the second best reformulation in our experiments. The best fully automatic reformulation used constraints derived from this new theorem:

$$\forall b c d ((b * c = d \wedge d * b = c) \rightarrow (c = d))$$

For QG5 quasigroups up to size five, on average, the solving time for the second best reformulation (i.e., the reformulation identified in [7]) took 77% of the time taken using the basic model. With the constraints derived from the above theorem, this time reduced to 71.7% that of the basic model.

³ A library of CSP benchmark problems (www.csplib.org).

5 Conclusions and Further Work

We have developed an automatic system, which uses machine learning and automated theorem proving to generate implied constraints for CSPs, thus improving the effectiveness of a CSP solver by reformulating the model of the problem. Using finite algebras as benchmark tests, in many cases, the time taken to reformulate problems represents a worthwhile investment, which is soon recovered when searching for solutions to larger problem instances. We have shown clear progress over the approach presented in [7] by (a) automating their semi-automated approach – which required much domain knowledge and constraint solving expertise – while recreating similar or better efficiency gains (b) successfully applying the procedure to new finite algebras, namely groups, Moufang loops and rings, the last of these being exceptional in that they have two algebraic operators rather than one, and (c) discovering new and effective implied constraints not identified in [7].

There is still room for improvement. In particular, in some cases, the system failed to create an improved solver model. There are a number of possible reasons for this, including:

- The basic model is the best possible. In this case, the approach would never improve the basic model.
- Improving constraints do exist but were not found by HR. We could counter this by running HR for longer or with heuristic guidance specifically tailored to this process. In addition, we plan to implement new ways for HR to make conjectures, for instance by allowing the user greater control over the concept formation process [18], or by directly noticing all-different properties, for which there are efficient solving methods.
- Improving constraints were found by HR but translated inefficiently. We have endeavoured to make the translation process as effective as possible, but we can improve this process. In particular, where possible, we aim to interpret conjectures as all-different constraints rather than sets of inequalities.
- Our method of assessing the effectiveness of constraints fails to identify the most useful constraints. In particular, a constraint may not be effective at low order but its effectiveness may increase as the domain size increases. In contrast, a constraint may be highly effective for small test sizes, but degrade as the domain size increases. For instance, the reformulation for QG6 quasigroups used the idempotency constraint ($\forall b (b * b = b)$), which applies to elements on the main diagonal of a multiplication table. As we see from tables 1 and 2, this achieves a time of around 12% of that for the basic model for problem sizes up to 6, but around 24% for sizes 9 and 10. Hence, we plan to implement techniques for extrapolating the effectiveness of constraints in order to choose them more intelligently.

We have recently extended the translation capabilities to enable the formulation of some first-order theorem proving problems as CSPs [5]. We also plan to generalise this approach further by enabling translation to the syntax of other solvers and extending into higher order logics. It is possible to use empirically plausible – but not necessarily proved – conjectures to define a case-split where two CSPs have to be solved in order to exhaust the search space, and we intend to pursue this possibility. Not only does this open up the possibility of distributing the solving process, but the sum of the times taken for the two cases may be less than the time taken to solve the basic model. Moreover, more complex case-splitting schemes can be used, which have already shown much promise in improving automated theorem proving techniques [12]. We also plan to study the ways in which concepts from HR can be used to streamline [11] constraint

solving. Here, the aim is to find a single solution, rather than all solutions, in which case generality can be sacrificed for efficiency by specialising a CSP. The ability to translate first-order CSPs is a useful tool outside of the application to CSP reformulation. We intend to use this for various applications, such as the comparison of model generators and CSP solvers for benchmark tests [5].

The study presented here adds weight to the argument that combining reasoning techniques can produce AI systems which are more than a sum of their parts. For instance, due to the limited number of examples at small domain sizes, the reformulations of QG6 quasigroups were based on conjectures found by HR using only two examples. Such little empirical evidence probably wouldn't be statistically significant in a straight machine learning exercise, and hence the conjectures might be ignored. However, because they were proved by Otter, they were as valid as ones with greater empirical evidence, which highlights the advantages to be gained by combining different reasoning approaches.

Acknowledgements

Ian Miguel is funded by a Royal Academy of Engineering / EPSRC research fellowship. We would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

- [1] C. Bessiere, R. Coletta, E.C. Freuder, and B. O'Sullivan. Leveraging the learning power of examples in automatic constraint acquisition. In *Proceedings of CP*, 2004.
- [2] C. Bessiere, R. Coletta, F. Koriche, and B. O'Sullivan. A SAT-based version space algorithm for acquiring constraint satisfaction problems. In *Proceedings of ECML*, 2005.
- [3] C. Bessiere, R. Coletta, and T. Petit. Acquiring parameters of implied global constraints. In *Proceedings of CP*, 2005.
- [4] M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In *Proc. Programming Languages: Implementations, Logics, and Programs*, 1997.
- [5] J. Charnley and S. Colton. Expressing general problems as CSPs. In *Proceedings of the Workshop on Modelling and Solving Problems with Constraints at ECAI*, 2006.
- [6] S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.
- [7] S. Colton and I. Miguel. Constraint generation via automated theory formation. In *Proceedings of CP*, 2001.
- [8] D. Duffy. *Principles of Automated Theorem Proving*. Wiley, 1991.
- [9] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM-AMS Proceedings 7, R Karp, ed.* pages 43–73, 1974.
- [10] A.M. Frisch, I. Miguel, and T. Walsh. CGRASS: A system for transforming constraint satisfaction problems. In *Proceedings of the Workshop on Constraint Solving and Constraint Logic Programming (LNAI 2627)*, pages 15–30, 2002.
- [11] C. Gomes and M. Sellman. Streamlined constraint reasoning. In *Proceedings of CP*, 2004.
- [12] F. Hoermann. Machine learning case splits for theorem proving. Master's thesis, Dept of Computing, Imperial College, London, 2005.
- [13] J. Little, C. Gebruers, D. Bridge, and E. Freuder. Capturing constraint programming experience: A case-based approach. In A. Frisch, editor, *International Workshop on Reformulating Constraint Satisfaction Problems at CP*, 2002.
- [14] J. Little, C. Gebruers, D. Bridge, and E. Freuder. Using case-based reasoning to write constraint programs. In *Proceedings of CP*, 2003.
- [15] K. Marriott and P. Stuckey. *Programming with Constraints: an Introduction*. MIT Press, 1998.
- [16] W. McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.
- [17] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [18] P. Torres and S. Colton. Applying model generation to concept formation. In *Automated Reasoning Workshop*, 2006.
- [19] M. Wallace. Practical applications of constraint programming. *Constraints*, 1(1/2):139–168, 1996.

Maintaining Generalized Arc Consistency on Ad-hoc n -ary Boolean Constraints

Kenil C. K. Cheng and Roland H. C. Yap¹

Abstract. Binary decision diagrams (BDDs) can compactly represent ad-hoc n -ary Boolean constraints. However, there is no generalized arc consistency (GAC) algorithm which exploit BDDs. For example, the global `case` constraint by SICStus Prolog for ad-hoc constraints is designed for non-Boolean domains. In this paper, we introduce a new GAC algorithm, `bddc`, for BDD constraints. Our empirical results demonstrate the advantages of a new BDD-based global constraint – `bddc` is more efficient both in terms of memory and time than the `case` constraint when dealing with ad-hoc Boolean constraints. This becomes important as the size of the ad-hoc constraints becomes large.

1 Introduction

Many real-life combinatorial problems such as scheduling can be modeled as a constraint satisfaction problem (CSP). There has been much attention on global constraints which make use of specialized consistency algorithms, for example `all_different` or `cumulative`. However, relatively less effort has been put into dealing with the *ad-hoc (n -ary) constraints*.

An ad-hoc n -ary constraint is defined explicitly either as a set of solutions or alternatively as a set of non-solutions. Obviously these tuples can be stored in an array or a table [3, 13]. The `table` constraint in ILOG Solver uses such an explicit implementation. A drawback of using a table is that the constraint is limited by the number of (non-)solutions. The problem is that the table size can grow exponentially with the arity of the constraint. This means that `table` constraints are restricted to either tight or loose constraints with moderate arity. In practice, large ad-hoc constraints can also be useful. For example, Cheng and Yap [7] demonstrate that improving a model for the Still-Life problem leads to large ad-hoc constraints, e.g. an ad-hoc constraint with 76 million solutions and 30 variables. Another drawback is the slow support checking within large tables. While this can be remedied by means of indexing [14], the additional data structure inevitably requires extra memory and manipulation effort.

One way to tackle the memory explosion problem is to identify or extract some arithmetic or symbolic relations from the solution set (e.g. [6, 10]). Although the final representation is usually more compact, they often need expensive pre-processing and the propagation on the higher-level representation could be weak.

Another approach is to represent the solution set in some compact data structure and build a tailor-made propagation algorithm (or combine existing propagators) on top of it (e.g. [1, 2, 8]). The ad-hoc non-binary `case` constraint provided by SICStus Prolog [17] belongs to this category. The `case` constraint also allows one to

specify the level of consistency (i.e. GAC or bounds consistency) to enforce. To use the `case` constraint, the solutions of the ad-hoc constraint should be represented as a directed acyclic graph (DAG) which is recursively defined as follows. A `case` DAG $G = \text{case}(x_i, [r_1 - G_1, \dots, r_m - G_m])$ defines the finite domain constraint

$$\llbracket G \rrbracket \equiv \bigvee_{k=1}^m (x_i \in r_k \wedge \llbracket G_k \rrbracket)$$

where each G_k is a `case` DAG and r_i specifies its bounding interval of integers. Figure 1 depicts the DAG representation of c_{dag}

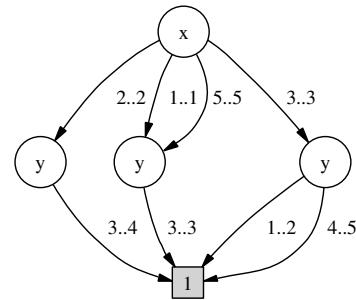


Figure 1. A DAG representation of c_{dag} .

with solutions $\{(x, 1), (y, 3)\}, \{(x, 2), (y, 3)\}, \{(x, 2), (y, 4)\}, \{(x, 3), (y, 1)\}, \{(x, 3), (y, 2)\}, \{(x, 3), (y, 4)\}, \{(x, 3), (y, 5)\},$ and $\{(x, 5), (y, 3)\}$. Each non-terminal node v (denoted by a circle) represents a variable x . An out-going edge of v has a label r that represents the constraint $x \in r$. The 1-terminal (in gray box) means T (true). The details of the consistency algorithm for `case` are not available. Beldiceanu [2] mentions that it traverses the `case` DAG in a depth-first fashion and updates the variable domains incrementally.

When every variable in a `case` constraint has a Boolean domain, the `case` DAG can also be represented by a binary decision diagram (BDD) [5]. BDDs are the state of the art representation for propositional logic in areas such as hardware verification [9]. Because of the success of BDDs for representing Boolean functions compactly, several authors have suggested to manipulate constraints with BDDs. Hawkins et. al. [12] solve CSPs involving set variables with BDDs. In [18] BDDs are used to represent the solution set in configuration problems. Cheng and Yap [7] show how to construct compact `case` constraints with BDDs and modeling with ad-hoc constraints to enhance propagation efficiency for the Still-Life problem. We conjecture BDD is also invaluable to many CSPs such as circuit fault

¹ National University of Singapore, 3 Science Drive 2, Singapore {chengchi,ryap}@comp.nus.edu.sg

diagnosis and balanced incomplete block designs² where Boolean constraints exist naturally.

Although it is always possible to model an ad-hoc Boolean constraint with the `case` constraint, one would expect a specialized consistency algorithm built on top of a BDD constraint representation to be more efficient – a simpler data structure and a less expensive implementation. Consequently, in this paper, we introduce `bddc`, a specialized version of `case` for Boolean domains. Like `case`, we choose to also traverse the BDD depth-first. To make this strategy computationally efficient, we avoid fruitless node visits with two techniques: Δ -cutoff and good/nogood recording. Roughly speaking, the former reduces the amount of nodes visited during the traversal while the latter makes the traversals less frequent. Although nogood recording [11] is an old technique for pruning the search space during search, it has not been applied much to consistency algorithms. We consider good recording as a generalization of support caching [15] for binary constraints. Another difference is that, our (no)goods are shared among identical constraints so that more savings on computation can be achieved.

In the next section we summarize our notations on CSP and BDD. The `bddc` algorithm is introduced in Section 3. Details on its implementation are provided. Section 4 presents and discusses our experimental results. We conclude in the last section.

2 Preliminaries

In this section, we present our terminology and a brief introduction to binary decision diagrams.

2.1 Constraint Satisfaction Problem

A *constraint satisfaction problem* (CSP) is a triple $\mathcal{P} = \langle X, \mathcal{D}, \mathcal{C} \rangle$, where $X = \{x_1, \dots, x_n\}$ is a set of *variables*, $\mathcal{D} = \{D_1, \dots, D_n\}$ is a set of *domains*, and \mathcal{C} is a set of *constraints*.³ Each variable x_i can only take values from its domain D_i , which is a set of integers. A *valuation* θ is a mapping of variables to integer values, written as $\{(x_1, d_1), \dots, (x_k, d_k)\}$. Let $vars$ be the function that returns the set of (free) variables appearing in a constraint or valuation. A k -ary constraint $c \in \mathcal{C}$ on an ordered set of k distinct variables is a subset of the Cartesian product $D_1 \times \dots \times D_k$ that restricts the values the variables in c can take simultaneously. Particularly, if $D_i = \{0, 1\}$ for all $1 \leq i \leq k$, then c is a *Boolean constraint*. The *scope* of c is $vars(c)$ and the *arity* of c is $arity(c) = k$. We say $T(F)$ is the trivially true (false) constraint. A valuation θ *satisfies* c , a.k.a. a *solution* of c , if and only if $\theta \in c$. Solving a CSP requires finding a value for each variable from its domain so that all constraints are satisfied.

Two constraints c_1 and c_2 are *equivalent*, denoted by $c_1 \equiv c_2$, if and only if they define the same relation, i.e. for any valuation θ , we have $\theta \in c_1 \iff \theta \in c_2$.

A constraint c is *consistent* if it is not equivalent to F . Otherwise, c is *inconsistent*. If $c \wedge x_i = d$ is consistent for each $x_i \in vars(c)$ and for each $d \in D_i$, then c is *generalized arc consistent* (GAC) [3, 16].

2.2 Binary Decision Diagrams

A *binary decision diagram* (BDD) [5] is a directed acyclic graph with two terminal nodes: the *0-terminal* which means F and the *1-*

² CSPLib: www.csplib.org

³ In this paper we use the terms “a set of constraints” and “a conjunction of constraints” interchangeably.

terminal which means T . Each non-terminal node u is labeled with a Boolean variable x_i and has two children G_1 and G_0 which are BDDs. Let $G = bdd(x_i, G_1, G_0)$ be the BDD rooted at u . Semantically G represents the Boolean constraint

$$[G] \equiv (x_i = 1 \wedge [G_1]) \vee (x_i = 0 \wedge [G_0]).$$

In BDD terms G_1 is the *1-successor* and G_0 is the *0-successor* of u . We call $[G]$ a *BDD constraint (defined by G)*. For brevity, we write G instead of $[G]$ when its meaning is clear. Figure 2 depicts a BDD which defines the Boolean constraint $(x_1 = 1 \wedge x_3 = 0) \vee (x_1 = 0 \wedge x_2 = 1 \wedge x_3 = 0)$. The solutions are $\{(x_1, 1), (x_2, 0), (x_3, 0)\}$, $\{(x_1, 1), (x_2, 1), (x_3, 0)\}$ and $\{(x_1, 0), (x_2, 1), (x_3, 0)\}$. The solid and the dotted out-going arrows from a node (in circle) point to its 1 and 0-successors respectively. The terminal nodes are drawn as gray boxes. Note that for defining a BDD constraint the presence of the 0-terminal is optional (as for the `case` constraint).

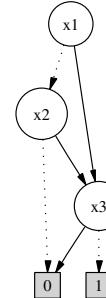


Figure 2. An example BDD.

A BDD constraint, `bddc`, is any BDD $G = bdd(x_i, G_1, G_0)$ that defines a Boolean constraint on variables x_i, \dots, x_k where $1 \leq i \leq k$. G_0 and G_1 are either the 0-terminal, the 1-terminal (when $i = k$), or a BDD $G' = bdd(x_j, G'_1, G'_0)$ where $i < j \leq k$. To ease the presentation and implementation, we require $j = i + 1$, i.e. any path from the root of G to the 1-terminal corresponds to a solution of $[G]$.

3 Maintaining GAC on a BDD Constraint

We now present the `bddc` algorithm which enforces GAC on a BDD constraint. The pseudo-code is given in Figure 3. The input is a BDD (constraint) G . The recursive call `bddc_r` (line 3) traverses G in a depth-first manner and creates the new domains of the variables in $vars(G)$. If G is inconsistent, `bddc_r` returns F and the solver backtracks (line 5). Otherwise, the domains of the variables are updated (line 7) to make G GAC. Good/nogood recording is implemented so that the expensive `bddc_r` is less frequently executed. Let θ be the current valuation of the subset of the variables in $vars(G)$. If `bddc_r` detects G is inconsistent, θ is inserted to *cache* as a *nogood* (line 4). On the other hand, if the new and the current domains are the same for every $x_i \in vars(G)$, θ is added to *cache* as a *good* (line 6) to avoid future consistency checks since the constraint is already GAC with those domains. The cache allows any subsequent call of `bddc` with the same θ , to either immediately trigger a backtrack if G is inconsistent (line 1) or just quit if G is already GAC (line 2).

Figure 4 gives the pseudo-code of `bddc_r`. It works as follows. Let $G = bdd(x_i, G_1, G_0)$. Recall that G defines the Boolean constraint

$$G \equiv (x_i = 1 \wedge G_1) \vee (x_i = 0 \wedge G_0).$$

```

bddc( $G$ )
begin
   $\theta :=$  the valuation of variables in  $vars(G)$ 
1   if  $cache[\theta] = F$  then fail //  $\theta$  is a nogood
2   if  $cache[\theta] = T$  then return //  $\theta$  is a good
   $visited := \emptyset$  // the set of visited nodes in  $G$ 
  foreach  $x_i \in vars(G)$  do
     $D'_i := \emptyset$  // init the new domain of  $x_i$ 
    // invariant:  $D'_i = D_i$  for all  $i \geq \Delta$ 
    // where  $D_i$  is the current domain of  $x_i$ 
     $\Delta := arity(G) + 1$ 
3    $ret := bddc\_r(G)$  // traverse  $G$  to find  $D'_i$ 's
  if  $ret = F$  then
    //  $G$  is inconsistent
     $cache[\theta] := F$  // nogood-recording
    fail // (solver) backtracks
  else
    //  $G$  is generalized arc consistent
    if  $\Delta = 1$  then
      //  $D_i = D'_i$  for all  $1 \leq i \leq arity(G)$ 
       $cache[\theta] := T$  // good-recording
    else
5     foreach  $x_i \in vars(G)$  where  $D'_i \neq D_i$  do
         $x_i \in D'_i$  // update the domain of  $x_i$ 
    end

```

Figure 3. Pseudo-code of bddc.

By induction, we can enforce GAC on G by enforcing GAC on G_d (via the recursive call of $bddc_r$ at line 9) for each d in the current domain D_i of x_i . If G_d is consistent, d is added to the new domain D'_i of x_i (line 10) and $bddc_r$ returns T . In the case there is no consistent successor, G is inconsistent and $bddc_r$ returns F . Finally, G is inserted to the cache $visited$ if it is consistent (line 12) and to the cache $pruned$ if it is not (line 13). These guarantee $bddc_r$ visits every node in G at most once. To make $bddc$ incremental, $pruned$ is not reset to empty at each call of $bddc$. This is because if G is inconsistent, it remains inconsistent when even more variables are assigned. By similar arguments, $visited$ must be zeroed initially.

More work can be saved by using the following optimization which we call Δ -cutoff. It utilizes the invariant (line 11)

$$D'_j = D_j \quad \forall j \geq \Delta$$

Thus, the call of $bddc_r(G_d)$ can be omitted whenever $i \geq \Delta$ and G_{1-d} is consistent (line 8). This is because G must then be consistent and all D'_j 's ($j \geq i$) are already “saturated” (namely, $e \in D_j \iff e \in D'_j$) and, by construction, they never shrink (i.e. once a value is put into D'_j , it will never be removed in subsequent recursive calls).

By induction on the input BDD constraint G , it is straightforward to show the following:

Theorem 1 *The bddc algorithm is sound, complete and always terminates.*

We now analyze the memory requirement for $bddc$. Consider the representation of the input BDD constraint G . Since G is static, we store its nodes in a fixed $n \times 2$ array (bdd) where n is the number of nodes in G . Then $bdd[u]$ refers to the BDD rooted at the

```

bddc_r( $G$ )
begin
  if  $G$  is the 0-terminal then return  $F$ 
  if  $G$  is the 1-terminal then return  $T$ 
  if  $G \in pruned$  then return  $F$  //  $G$  is inconsistent
  if  $G \in visited$  then return  $T$  //  $G$  is GAC
  let  $G = bdd(x_i, G_1, G_0)$ 
  // recall  $G \equiv (x_i = 1 \wedge G_1) \vee (x_i = 0 \wedge G_0)$ 
   $ok_1 := F$  // assume  $G_1$  is inconsistent
   $ok_0 := F$  // assume  $G_0$  is inconsistent
  foreach  $d \in D_i$  (the current domain of  $x_i$ ) do
    if  $ok_{1-d} \equiv T$  and  $i \geq \Delta$  then
      //  $G_{1-d}$  is consistent (so is  $G$ )
      // also, the old and the new domains
      // of  $x_j$  ( $j \geq i$ ) are the equal
      break
    // otherwise, traverse  $G_d$  recursively
     $ok_d := bddc\_r(G_d)$ 
    if  $ok_d \equiv T$  then
      //  $G_d$  is consistent
      // add  $d$  to the new domain of  $x_i$ 
       $D'_i := D'_i \cup \{d\}$ 
      if  $i = \Delta - 1$  and  $D_i = D'_i$  then
         $\Delta := \Delta - 1$  // update the invariant
    if  $ok_0 \equiv T \vee ok_1 \equiv T$  then
      //  $G$  is visited and  $G$  is consistent
       $visited := visited \cup \{G\}$ 
      return  $T$ 
    else
      //  $G$  is inconsistent
       $pruned := pruned \cup \{G\}$ 
      return  $F$ 
  end

```

Figure 4. Pseudo-code of $bddc_r$.

node u (represented as an integer) and $bdd[u][d]$ points to the d -successor of u . The size of the array is $2n[\log_2 n]$ bits, where $[x]$ is the smallest integer larger than x .

The caches $visited$ and $pruned$ are represented as bit vectors of length n . The u -th bit is set to 1 if and only if the BDD rooted at u is in the cache. Note that $bddc$ will be invoked at most $k = arity(G)$ times, whenever a variable in its scope is instantiated. For trailing purpose, the call of $bddc$ at time t must make its copy of $pruned$ updated at time $t - 1$. As a result, there will be up to k copies of $pruned$, and at most $(k + 1)n$ bits are required for the two caches.

We implement $cache$ as a hash table of fixed size h (i.e. no collision resolution). A (no)good θ is encoded as two bit vectors, namely α_0 and α_1 , of length k . The i -th bit of α_d is set to 1 if and only if $(x_i, d) \in \theta$. Another 1 bit is used to identify whether θ is a good or a nogood. The size of $cache$ is therefore $(2k + 1)h$ bits.

Often a CSP consists of several identical constraints on different variables. For example, it is common to have a disequality constraint between every two variables. Since a BDD constraint could be exponentially large, it is especially advantageous to let them share as many data structures as possible. In our implementation of $bddc$, besides the obvious bdd (which is static) and $visited$ (which is basically a local variable), $cache$ is also shared. This is correct because

the encoded valuations are variable independent. The shared *cache* is actually more powerful. It can cut down more fruitless computations – a (no)good detected by one constraint can be reused by all other identical constraints. Theorem 2 summarizes the memory usage:

Theorem 2 Suppose there are m identical k -ary Boolean constraints defined by a BDD G with n nodes, and assume there are h slots in cache. The overall memory requirement for the m constraints is $\Theta((2\lceil \log_2 n \rceil + 1 + mk)n + (2k + 1)h)$ bits.

Note that the $(2k + 1)h$ bits are needed only when good/nogood recording is enabled, and less memory can be used if we restrict the arity of (no)goods. Actually, the memory requirement for bddc are rather low. The alternative of representing the s solutions in a table alone uses at least sk bits and we would expect when BDDs are applicable that $s \gg n$. Extra memory is necessary for trailing.

4 Experimental Results

In this section, we report the runtime performance and memory usage of bddc. We used SICStus Prolog 3.12.3 [17] as our experimental and evaluation platform because it is the only constraint system with an implementation of the *case* global constraint. It would not be fair to compare with a more explicit ad-hoc constraint representation like *table* simply due to the space and time overheads of large tables. The *case* constraint in SICStus Prolog being a generalized version of bddc serves as a more meaningful benchmark.

Our implementation of the bddc solver is in C and uses the built-in global constraint APIs as well as the C-Prolog interface in SICStus Prolog. Experiments were run on a PC running Windows XP, with a P4 2.6 GHz CPU and 1 GB physical memory.

We have experimented with two sets of benchmarks of 25 instances each. One set contains many small BDD constraints (BDD size $\approx 3K$) while the other is the opposite; large BDD constraints (BDD size $\approx 22K$) but fewer constraints. Thus, the intention is to investigate the performance of bddc with respect to the two major scaling factors: the number of constraints and the size of an ad-hoc constraint.⁴ An instance in the benchmark is described using the notation, (v, m, k, p) , which denotes a problem with v Boolean variables and m copies of a random k -ary BDD constraint on different (randomly chosen) subset of variables. The tightness p means the BDD constraint has about $(1 - \frac{p}{100}) \times 2^k$ solutions. The values of m and p were chosen by trial-and-error such that each benchmark has both satisfiable and unsatisfiable instances, and the instances are neither too simple nor too difficult to solve. The BDD is generated in a depth-first, post-order manner where identical sub-BDDs are merged and the terminal nodes are chosen randomly based on p .

To compare the efficiency of the bddc solver, it is necessary to take into account the instantiation order from different search/labeling strategies. Intuitively, as we will see, one would expect a difference between a top-down and bottom-up search strategy. We have evaluated the following four instantiation orderings (*ord*) on a bddc constraint as follows. *Top-down* (*TD*) means the natural, bddc lexicographic order. *Bottom-up* (*BU*) is the reverse of *TD*. *Zip-zap* (*ZZ*) interleaves *TD* and *BU* (i.e. $x_1, x_k, x_2, x_{k-1}, \dots, x_{\frac{k}{2}}$). *Middle-out* (*MO*) reverses *ZZ*.

Table 1 gives the results on the first benchmark (21, 2713, 15, 79) with many small bddc constraints. The figures given are the mean

⁴ Our CSP instances have significantly more constraints than the ones used in a recent paper [14] about the ILOG *table* constraint. The arity and the size of the constraints in the two papers are comparable.

of the results obtained over the 25 instances. Each instance has 2713 identical 15-ary BDD constraints with 3576 nodes on average. The column *t* (in seconds) gives the search time⁵ for a solution of an instance, when the BDD constraints are implemented with bddc. The column *̄n* lists the average number of nodes visited during a BDD traversal. This can be computed as

$$\bar{n} = \frac{\text{total number of } \text{bddc_r} \text{ called during search}}{\text{total number of } \text{bddc} \text{ called during search}}.$$

The last column gives the search time when *case* is used instead. We highlight the fastest execution time in **bold**. The search times given here do not include the time for initialization. On average *case* takes 4.8 seconds to initialize in the first benchmark (with small BDDs) and 14.5 seconds in the second one (with large BDDs). In both tests, bddc builds its own data structures within 1 second. The number of backtracks is given in the column *bt*. During search the zero branch (i.e. $x_i = 0$) is always explored first. We experimented with three variations of bddc as follows. *DF* is a basic depth-first algorithm which does not have the good/nogood recording nor Δ -cutoff. *DF + Δ* uses only Δ -cutoff. *DF + Δ + c* is the full version of bddc as in Figures 3 and 4. There are 2^{20} slots in *cache*. Each cache entry is defined as *unsigned long long* (8 bytes) and hence the memory used for *cache* is 8 MB.

<i>ord</i>	<i>bt</i>	<i>DF</i>		<i>+Δ</i>		<i>+Δ + c</i>		<i>case</i>
		<i>t</i>	<i>̄n</i>	<i>t</i>	<i>̄n</i>	<i>t</i>	<i>̄n</i>	
<i>TD</i>	2131	77.2	111.5	45.9	36.5	24.5	0.3	90.8
<i>BU</i>	2144	298.9	681.9	81.2	129.4	25.0	1.2	236.5
<i>ZZ</i>	2150	171.6	323.4	61.7	68.8	26.0	0.9	141.7
<i>MO</i>	1503	102.5	280.9	43.9	68.4	18.2	1.0	118.1

Table 1. Experimental results on (21, 2713, 15, 79).

Our first observation is that, to enforce (restore) GAC on a BDD constraint after a variable assignment, on average *DF* visits only 111.5 nodes when the variables are assigned top-down, but 681.9 nodes when the labeling is done bottom-up. Notice that *̄n* is independent to the exact search tree because bddc is called only at each node of the search tree and by definition *̄n* is the average number of *bddc_r* invoked in a bddc call. As a result, despite the number of backtracks under *TD* and *BU* which are more or less equal, the search is 2.9 times slower under *BU* since *BU* leads to more nodes visited.

The explanation for the variations in *̄n* is as follows. When the variables are assigned top-down, the BDD shrinks (implicitly) and by definition, the valuation leads to the 0-terminal if the constraint is inconsistent. Consequently *DF* can detect inconsistency or enforce GAC within a few node visits. On the other hand, under the bottom-up labeling order, *DF* must then traverse a large (upper) part of the BDD to reach the assigned variables at the bottom; also, the BDD does not necessarily reduce in size with the number of assigned variables. We can see that *TD* and *BU* correspond respectively to the best and the worst scenarios for *DF*. For similar reason, *DF* visits less nodes under *MO* than *ZZ*. Compared with *case*, *DF* is faster under *TD* and *MO* but slower under *BU* and *ZZ*.

Given that ad-hoc constraints may be rather large, memory usage may actually be a more relevant factor than solver efficiency. Here,

⁵ Time was reported by the built-in predicate *statistics* with parameter *runtime*.

we can see the effect of the specialized `bddc` on memory usage. On average `bddc` uses 32.5 MB while the average memory usage for `case` is 728.3 MB.⁶ Note that in both cases, we are already exploiting the use of a directed acyclic graph representation, and the difference shows the contrast between `bddc` and `case`. We also see that for `case`, the memory usage is getting close to the maximum physical memory on our PC. In other words, the benchmark size seems to be nearing the limits of `case`.

Δ -cutoff drastically reduces the computation effort of `bddc`. $DF + \Delta$ visits only 19 (*BU*) to 33 (*TD*) percent of nodes visited by DF . It is particularly effective when the variables are assigned bottom-up since there is a higher chance to “saturate” the domains of the (instantiated) variables at the bottom and push up the cutoff level (Δ). This makes $DF + \Delta$ up to 2.7 times faster than DF .

Since there are 2713 identical BDD constraints in an instance, the use of good/nogood recording is expected to cut down the amount of wasteful work by a huge margin. Our empirical results show that, every call of $DF + \Delta + c$ visits up to 1.2 nodes on average. In the case of *TD*, it is just 0.3 node on average. The cache usage is very effective with up to 43000 cache slots filled and a cache hit rate of around 99 percent. As a consequence, representing the BDD constraints with `bddc` instead of `case` speeds up the search by 2 to 9 fold approximately. There is a dramatic improvement in the time for *BU* which speeds up about 12 fold. Thus, caching is important in making the `bddc` solver more robust – the execution time is now fairly independent of the labeling order.

The results on the second benchmark (21, 133, 18, 78) with fewer but larger constraints are summarized in Table 2. There are 133 identical BDD constraints. The arity is now increased to 18 and each BDD constraint has 22575 nodes on average. The memory usage for `case` is 166.8 MB and that for `bddc` is 30.4 MB.

ord	bt	DF		$+\Delta$		$+\Delta + c$		<code>case</code>
		t	\bar{n}	t	\bar{n}	t	\bar{n}	
<i>TD</i>	17768	43.1	81.6	31.3	33.7	11.2	1.8	53.7
<i>BU</i>	16379	155.0	651.8	54.7	160.2	12.3	10.9	217.7
<i>ZZ</i>	16670	91.5	310.2	40.6	82.5	12.1	7.4	99.3
<i>MO</i>	14657	67.7	242.4	37.3	77.0	11.5	6.5	98.9

Table 2. Experimental results on (21, 133, 18, 78).

Although the statistics in the two benchmarks share similar patterns, there are two subtle differences. First, Δ -cutoff is slightly less effective in the second benchmark. This is because when the arity is large, the cutoff level becomes relatively low, even if the absolute value is unchanged. Second, as the instances are more difficult to solve (i.e. more backtracks) and there are less identical constraints, the efficiency of good/nogood recording degrades. Our cache log reports up to 151114 slots are filled and the cache hit rate is more or less 90%. On the other hand, `bddc` scales well. The full version is about 4 to 17 times faster than `case`. Now even the naive DF outperforms `case`. Our results demonstrate the effectiveness of the full `bddc` algorithm in terms of runtime and memory.

⁶ The memory used by SICStus Prolog was measured by `statistics` with parameter `memory`. For `bddc`, the memory allocated in C was calculated based on Theorem 2, except that in actual the implementation uses `unsigned long bdd[] []` and `unsigned long long cache[]`.

5 Concluding Remarks

We have proposed a new algorithm, `bddc`, which enforces GAC on ad-hoc n -ary Boolean constraint defined in a BDD. Our experimental results show `bddc` always outperforms `case` in terms of memory usage and computation efficiency. This justifies the need for a tailor-made GAC algorithm on BDD constraint particularly as the size of the ad-hoc constraint becomes large.

The implementation techniques we have presented in this paper are not restricted to `bddc` – Δ -cutoff can as well be used to prune unnecessary node visits during the traversal of a `case` DAG; also, as suggested in [15], caching should be helpful for arbitrary consistency algorithms, especially for the computationally expensive ones.

One unaddressed issue is to enforce GAC on a conjunction of (BDD) constraints (e.g. [4]). Working on ad-hoc constraints directly opens up new possibilities to tackle this problem, for example, by combining several constraints into a single one (e.g. [7]). We also plan to investigate the more general non-boolean case.

ACKNOWLEDGEMENTS

This paper was written while Roland Yap was visiting the Swedish Institute of Computer Science and their support and hospitality are gratefully acknowledged.

REFERENCES

- [1] R. Barták, ‘Filtering algorithms for tabular constraints’, in *Colloquium on Implementation of Constraint and Logic Programming Systems (CICLOPS)*, pp. 168–182, (2001).
- [2] N. Beldiceanu, ‘Global constraints as graph properties on structured networks of elementary constraints of the same type’, TR T2000-01, SICS, (2000).
- [3] C. Bessière and J.-C. Règine, ‘Arc consistency for general constraint networks: Preliminary results’, in *IJCAI*, pp. 398–404, (1997).
- [4] C. Bessière and J.-C. Règine, ‘Local consistency on conjunctions of constraints’, in *ECAI’98 Workshop on Non-binary Constraints*, pp. 53–59, (1998).
- [5] R. E. Bryant, ‘Graph-based algorithms for Boolean function manipulation’, *IEEE Trans. on Comp.*, **35**(8), 667–691, (1986).
- [6] K. C. K. Cheng, J. H. M. Lee, and P. J. Stuckey, ‘Box constraint collections for adhoc constraints’, in *CP*, pp. 214–228, (2003).
- [7] K. C. K. Cheng and R. H. C. Yap, ‘Ad-hoc global constraints for Life’, in *CP*, pp. 182–195, (2005).
- [8] K. C. K. Cheng and R. H. C. Yap, ‘Constrained decision diagrams’, in *AAAI*, pp. 366–371, (2005).
- [9] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, The MIT Press, 1999.
- [10] T.B.H. Dao, A. Lallouet, A. Legtchenko, and L. Martin, ‘Indexical-based solver learning’, in *CP*, pp. 541–555, (September 2002).
- [11] R. Dechter, ‘Learning while searching in constraint satisfaction problems’, in *AAAI*, pp. 178–185, (1986).
- [12] P. J. Hawkins, V. Lagoon, and P. J. Stuckey, ‘Solving set constraint satisfaction problems using ROBDDs’, *JAIR*, **24**, 109–156, (2005).
- [13] P. D. Hubbe and E. C. Freuder, ‘An efficient cross product representation of the constraint satisfaction problem search space’, in *AAAI*, pp. 421–427, (1992).
- [14] O. Lhomme and J.-C. Règine, ‘A fast arc consistency algorithm for n-ary constraints’, in *AAAI*, (2005).
- [15] C. Likitvivatanavong, Y. Zhang, J. Bowen, and E. C. Freuder, ‘Arc consistency in MAC: A new perspective’, in *1st Intl. Workshop on Constraint Propagation and Implementation*, (2004).
- [16] A. K. Mackworth, ‘On reading sketch maps’, in *IJCAI*, pp. 598–606, (1977).
- [17] Swedish Institute of Computer Science, *SICStus Prolog User’s Manual*.
- [18] E. van der Meer and H. R. Andersen, ‘BDD-based recursive and conditional modular interactive product configuration’, in *CSPIA Workshop*, pp. 112–126, (2004).

A Study on the Short-Term Prohibition Mechanisms in Tabu Search

Luca Di Gaspero¹ and Marco Chiarandini² and Andrea Schaerf¹

Abstract. Tabu Search (TS) is a well known local search method which has been widely used for solving AI problems. Different versions of TS have been proposed in the literature, and many features of TS have been considered and tested experimentally. The feature that is present in almost all TS variants is the so called (short-term) tabu list which is recognised as the crucial issue of TS. However, the definition of the parameters associated with the tabu list remains in most TS applications still a handcrafted activity.

In this work we undertake a systematic study of the relative influence of few relevant tabu list features on the performances of TS solvers. In particular, we apply statistical methods for the design and analysis of experiments. The study focuses on a fundamental theoretical problem (GRAPH COLOURING) and on one of its practical specialisation (EXAMINATION TIMETABLING), which involves specific constraints and objectives. The goal is to determine which TS features are more critical for the good performance of TS in a general context of applicability.

The general result is that, when the quantitative parameters are well tuned, the differences with respect to qualitative parameters become less evident.

1 INTRODUCTION

Local search is a search paradigm which has evidenced to be very effective for a large number of AI problems [18]. Tabu Search (TS) [15] is one of the most successful local search methods, which has been widely used for solving AI problems. Different versions of TS have been proposed in the literature, and many features of TS have been considered and tested experimentally. They range from long-term tabu, to dynamic cost functions, to strategic oscillation, to elite candidate lists, to complex aspiration criteria, just to mention some (see [15] for an overview).

The feature that is included in virtually all TS variants is the so called (*short-term*) *tabu list*. The tabu list is indeed recognised as the basic ingredient for the effectiveness of a TS-based solution, and its behaviour is a crucial issue of TS.

Despite the importance of a correct empirical analysis, which has been emphasised in the general context of heuristic methods [2, 19] and even in the specific case of TS [28], the definition of the parameters associated with the tabu list remains in most research work still a handcrafted activity. Often, the experimental work behind the parameter setting remains hidden or is condensed in a few lines of text

¹ DIEGM, University of Udine, via delle Scienze 208, I-33100, Udine, Italy, email: l.digaspero@uniud.it||schaerf@uniud.it

² Department of Mathematics & Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M – Denmark, email: marco@imada.sdu.dk

reporting only the final best configuration. Even the recently introduced *racing* methodology for the tuning of algorithms [4] only allows to determine the best possible configuration. These procedures are certainly justified from a practical point of view, but a more detailed description of the behaviour and sensibility of the algorithm with respect to its different factors and parameters is surely of great interest in the research field.

In this work, we aim at determining which factors of basic TS are important and responsible for the good behaviour of the algorithm and its robustness (i.e., the capability of exhibiting a stable behaviour across several set of instances). In contrast to the one-factor-at-a-time approach used in [28], we use different experimental design techniques [23], with parametric and non-parametric analysis of variance and racing algorithms. The suitability and comparison of these techniques in the field of search algorithms is still an open issue, hence we use more than one method to provide complementary support to their conclusions. We focus the analysis on both a fundamental theoretical problem, namely the GRAPH COLOURING problem, and on one of its practical specialisation, the EXAMINATION TIMETABLING problem, for which there is a consistent literature and many benchmark instances.

Although the statistical methods applied indicate that an algorithmic configuration of general validity does not arise, they point out that for these problems TS is robust enough w.r.t. the qualitative features, and thus it is reasonable to focus mainly on tuning the quantitative parameters. In addition, we strengthen our analysis with a comparison with previous work thus providing absolute validity to the results here presented and avoiding the pitfall of a self-standing work. We show indeed that the so-configured solution algorithm produces results that are comparable with the state-of-art.

2 PROBLEM DEFINITIONS

The first problem we consider is the classical GRAPH (VERTEX) COLOURING problem in its chromatic number formulation [14]. Given an undirected graph $G = (V, E)$, the GRAPH COLOURING problem consists in finding the minimal number of colours such that there exists an assignment of exactly one colour to each vertex and vertices connected by an edge $e \in E$ receive different colours.

The EXAMINATION TIMETABLING problem consists in scheduling the exams of a set of university courses in rooms and timeslots. In doing this, overlaps of exams with students in common must be avoided, exams must be fairly spread for each students, and room capacity constraints must be satisfied [26]. Among the different EXAMINATION TIMETABLING versions, we focus on the formulation proposed by Burke *et al.* [5]. According to this formulation, the examination session consists of a given fixed number of periods, and each period belongs to a specific day. Exams with common students

cannot be scheduled in the same period (first order conflicts), but they should also be not scheduled in adjacent periods of the same day (second order conflicts). It is easy to see, that this formulation corresponds to the GRAPH COLOURING problem in which the exams represent the vertices, the periods the colours, and two nodes are connected if they have students in common. The differences stem from the fact that for the EXAMINATION TIMETABLING the number of periods is fixed, whereas the objective function consists in the second order conflicts (weighted by the number of students).

3 TABU SEARCH BASICS

Local search is based on the idea of navigating a space of solutions (called *search space*) by iteratively moving from one state to one of its “neighbours”, which are obtained by applying a simple local change to the current solution. Several criteria for the selection of the move can be defined upon this idea. In particular, TS explores the full current neighbourhood and selects, as the new current state, the neighbour that gives the minimal value of a cost function, independently of whether its cost is less or greater than the current one. This selection allows the search to *escape* from local minima, but creates the risk of cycling among a set of states. In order to prevent cycling, TS uses a prohibition mechanism based on memory of past iterations: a list (called tabu list) stores the most recently accepted moves, so that the *inverses* of the moves in the tabu list are forbidden (i.e., the moves that are leading again toward the just visited local minimum).

The precise instantiation of the above general scheme can be done in different ways. In details, the three main features related to the tabu list are the following:

Neighbourhood exploration: Often exploring the full neighbourhood and determining the best move can be computationally too expensive for a tabu search algorithm to perform well. The exploration of the neighbourhood is therefore reduced to a part of it. The part of the neighbourhood explored can be determined probabilistically, heuristically or through considerations on structural properties of the problem that guarantee no improvement for the neighbours omitted.

Prohibition power: The prohibition power determines which moves are prohibited by the fact that a move is in the tabu list. A move is normally composed of several attributes; depending on the prohibition power, the tabu status can be assigned only to the move with the same values for all attributes or to the set of moves that have one or more attribute equal.

List dynamics: The list dynamics determines for how many iterations a move remains in the tabu list. This can be a fixed value, or a value selected randomly in an interval, or a value selected adaptively on the basis of the current state of the search.

In the next section, we propose different alternatives for the definition of the above features in the two case studies.

4 INSTANTIATION OF TABU SEARCH FEATURES

The search space considered is the same for both problems and is composed by one variable for each vertex (resp. exam) whose domain corresponds to the set of colours (resp. periods). The neighbourhood of a state is composed by all states reachable by changing exactly one colour at a vertex. In this setting, a local search move m

can be identified by three attributes: a vertex v , its old colour o and its new colour n and it can be identified by the triple $m = \langle v, o, n \rangle$. Note that the two attributes v and n suffice to identify the move uniquely.

The exploration of the neighbourhood can be restricted to only those vertices involved in a conflict (i.e., adjacent vertices that are assigned the same colour). In this way, it is guaranteed that no improving move is missed and, although the worst case complexity remains $O(kn)$, in practise we obtain a large reduction of search effort especially when the search explores states that are close to good solutions. This reduction is very helpful, and we do it in all cases. A further possible reduction of the neighbourhood exploration could be achieved by means of the MinConflict heuristic [22]. Consequently, we decide to implement the following three neighbourhood exploration reductions:

No reduction (NoRed): for each vertex involved in a conflict all the colours are tried and the overall best move is taken.

MinConflict Vertex (MinConf_v): for a vertex randomly chosen among those involved in a conflict the colour that yields the best move is chosen.

MinConflict Colour (MinConf_c): for all vertices involved in a conflict a colour is selected at random and the best overall move is taken.

For the prohibition power, assuming that the move $\langle v, n, o \rangle$ is in the tabu list, we consider the following three alternatives (where the underscore means “any value”):

Strong: All moves of the form $\langle v, _, _ \rangle$ are prohibited.

Medium: All moves of the form $\langle v, _, o \rangle$ are prohibited

Weak: Only the single move $\langle v, n, o \rangle$ is prohibited

In other terms, in the first case, it is not allowed to recolour the vertex in the tabu iterations. In the second case, it is not allowed to recolour the vertex with its old colour. In the third case, it is not allowed only to make the reverse of the tabu move.

Finally, in regards to the list dynamics, we also consider two possibilities:

Interval: The size of the list can vary in the range $[t_b, t_b + w]$ of width w . Each accepted move remains in the list for a number t of iterations equal to $t_b + \text{Random}(0, w)$, where $\text{Random}(a, b)$ is the uniform integer-valued random distribution in $[a, b]$

Adaptive: The value t depends on the current state. We use the formula (proposed in [13]) $t = \lfloor \alpha * c_s \rfloor + \text{Random}(0, t_b)$ where t_b and α (real value) are parameters, and c_s is the number of conflicts in the current state s .

A special case of Interval is the case, called Fixed, in which $w = 0$, so that the tabu list is a queue of size t_b . At each iteration, the accepted move gets in and the oldest one gets out. All moves remain in the list for exactly t_b iterations.

As a final remark, it is worth to note that according to the common practise in Tabu Search applied to the GRAPH COLOURING problem (e.g., [16]), the issue of finding the minimum number of colours is solved as a sequence of decision problems. The procedure stops and returns the value k , as soon as no solution with $k - 1$ colours can be found.

5 EXPERIMENTAL METHODOLOGY

It is recognised that studying local search algorithms in an analytical way is a complex task which often yields results with scarce practical

implications [17, 24]. The assessment of these algorithms is therefore carried out through empirical tests and the experimental design theory, largely applied in the engineering fields, provides the guidelines to do this in a methodological way [25, 3, 8].

The tabu search features introduced above are *treatment factors* (i.e., aspects that are hypothesised to have an effect on the behaviour of the algorithm) and they can be subdivided in two types: quantitative and qualitative. The three qualitative factors are the neighbourhood exploration, the prohibition power and the list dynamics, and consist each of different levels. The quantitative factors are the numerical parameters of the list dynamics strategy and may assume an unlimited number of values. There are two issues that complicate the factorial design: (i) the qualitative parameters do not cross with all other factors (for example, there is no α parameter with Interval list dynamics); (ii) the importance of each of the two qualitative factors strongly depends on the values assigned to the underlying quantitative parameters.

We follow two approaches in the analysis: a *bottom-up* approach consisting in first studying the best configurations of quantitative parameters for each combination of qualitative factors and then analysing the effects of the latter; and a *top-down* approach consisting in analysing the effects of the qualitative factors in a first step and then refining the analysis on the quantitative parameters for the best configuration. Both these approaches have a drawback: the bottom-up approach may require a huge amount of experiments and analysis; the top-down approach may be biased by the few values chosen for the qualitative parameters.

A further issue is the level of the analysis to be undertaken. Clearly discriminating results on single instances is not meaningful in practise; however, both an aggregate analysis and a separated analysis on single classes of instances are relevant for practical applications. We include, therefore, in the case of GRAPH COLOURING six classes of instances³ determined by the type of random graphs (geometric and uniform) and edge density (0.1, 0.5, 0.9). We maintain, instead, the size of the graph fixed to 1000 vertices.

Finally, a matter of concern in the statistical analysis of algorithms is the choice between parametric and non-parametric methods. We opt even in this case for maintaining both approaches. Indeed, being based on two different transformations of the results (normalisation and ranking) the two approaches are likely to produce a different statistical inference. From a theoretical perspective, the parametric analysis of variance through the F -ratio (ANOVA) is a procedure which has been recognised as particularly robust against deviation from the assumption of normal distribution of data [23]. The non-parametric analysis of variance through the Friedman test [9] by ranks is instead grounded on less assumptions but is unable to study interactions between factors.

6 ANALYSIS

We split our analysis in two parts, one for each problem, and we report the results on the two problems separately.

6.1 Graph Colouring

The experimental designs and the values considered for the factors of analysis are reported in Table 2a. We consider the random seed to generate different instances as a blocking factor [10]. The distinction between two designs, \mathcal{D}_A and \mathcal{D}_B , is necessary because of

³ Instances are newly generated using the Culberson's generator available at www.cs.alberta.ca/~joe/Coloring

the different meaning of parameter t_1 . Note that the degenerate case $t_1 = w = 0$ in Interval list dynamics corresponds to a fixed tabu length depending only on the parameter $t_2 = t_b$.

For each algorithmic configuration we collect one run on each instance. The response variable is the minimal number of colours found in a run after 10 million algorithm iterations. In the parametric analysis the number of colours $r(i)$ obtained by each algorithmic configuration on an instance i is normalised by the formula $e(r, i) = \frac{r(i) - r_{best}(i)}{r_{RLF}(i) - r_{best}(i)}$, where r_{best} is the best result achieved on instance i by all algorithms and r_{RLF} is the average solution produced by the RLF heuristic [20], which is used as the starting solution for the tabu search. Therefore, we consider $e(r, i)$ as our response variable. In the non-parametric analysis, instead, results are ranked within instances and no normalisation is needed. A level of significance of 5% is maintained over all the statistical tests.

In the top-down approach we maintain the two designs \mathcal{D}_A and \mathcal{D}_B separated. A parametric analysis of variance (ANOVA) by means of the F -ratio provides a general view of the significant effects. ANOVA assumes a linear model of responses and treatments. We use an automated stepwise model selection based on AIC [1] to distinguish which treatment combinations are significant enough to be included in the model. Accordingly, all main effects and interactions between two factors result significant. The analysis should therefore discriminate too much and this approach is not helpful. It is important to remark however that also the two instance class factors interact with the algorithm factors hence indicating that results might vary between instance classes.

In the bottom-up approach we use a sequential testing methodology for tuning of the parameters t_1 and t_2 . In particular we adopt the RACE algorithm proposed by Birattari [4] based on t -test for multiple comparisons and Holm's adjustment of the p-value. This procedure determines the best t_1 and t_2 for each combination of the other factors and allows to remove the quantitative parameters from the analysis. What remains is a unique design \mathcal{D} of 5 qualitative factors that can be analysed by ANOVA. Interaction with instance type and density is significant and indicates that a fine grained analysis should discriminate among the instance classes. We decide to focus, however, primarily on robust results, i.e., settings that are robust across multiple classes of instances. Hence, simplifying the analysis we remain with 3 algorithmic factors and study their main effects and interactions. In Table 3a and Figure 1 we report the results of this analysis. The most important result, emphasised from the visualisation of the analysis through interaction plots in Figure 1, is the bad performance of the MinConf neighbourhood exploration strategy on the colour. Another ineffective choice is the employment of a Strong prohibition power.

The non-parametric analysis through the Friedman analysis of variance followed by its extension in to multiple comparisons [9] confirms these results. In these tests, the factors studied are the combination of neighbourhood exploration, prohibition power and list dynamics (hence, a unique factor with $3 \times 4 \times 2$ levels) and the instance blocking factor. The conclusions above are recognised by the fact that all levels composed by the MinConf_c neighbourhood exploration strategy or Strong prohibition power are statistically inferior to the others.

6.2 Examination Timetabling

For space limits, we restrict the presentation of results of the bottom-up analysis only. Given the results on the GRAPH COLOURING we consider only the NoRed neighbourhood. The values of the param-

Table 1: Experimental designs employed in the experimentation for the two problems

Factors	Design \mathcal{D}_A	Design \mathcal{D}_B
Qualitative Neighbourhood Prohibition power List dynamics	{NoRed, MinConf _v , MinConf _c } Interval	{Weak, Medium, Strong} Adaptive
Quantitative $t_1=w/\alpha$ $t_2=t_b$	{0, 5, 10} {5, 10, 15, 20}	{0.3, 0.6, 0.9} {5, 10, 20}
Blocking Type Density Random seed	{Uniform, Geometric} {0.1, 0.5, 0.9} [1, ..., 10]	

(a) Experimental designs for GRAPH COLOURING

Factors	Design \mathcal{D}_A	Design \mathcal{D}_B
Qualitative Neighbourhood Prohibition power List dynamics	{NoRed}	{Weak, Medium, Strong}
Quantitative $t_1=w/\alpha$ $t_2=t_b$	Interval {0, 5, 10} {5, 10, 15, 20, 25}	Adaptive {0.3, 0.5, 0.8} {5, 10, 20, 30}
Blocking Instances		Carter's data-set [7]

(b) Experimental designs for EXAMINATION TIMETABLING

Table 2: The p-values corresponding to the ANOVA F-ratio in the bottom-up analysis performed on the two problems: quantitative parameters have been set by the RACE procedure; blocking factors are not taken into account.

Main Effects	Pr(> F)	Interaction Effects	Pr(> F)
Prohibition power	< 10 ⁻¹⁵	Prohibition Neighb.	< 10 ⁻¹⁵
Neighbourhood List Dynamics	< 10 ⁻¹⁵ 0.5971	Prohibition List dyn. Neighb. List dyn.	0.6450 0.1909

(a) Numerical results of ANOVA performed on GRAPH COLOURING

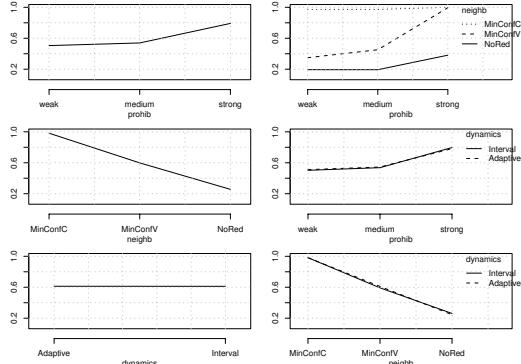


Figure 1: Interaction plots for the factors of the bottom-up analysis on the GRAPH COLOURING problem.

eters tested are shown in Table 2b.

The RACE algorithm is used for the selection of the quantitative parameters. Experiments are carried out on 7 instances taken from Carter's data-set [7]. Each run of a configuration was granted 120 seconds of CPU time on an AMD Athlon 1.5GHz computer running Linux. The outcomes of the RACE procedure are reported in Table 3 that shows the configurations for which no significant difference was found after 50 stages.

Table 3: RACE results for EXAMINATION TIMETABLING

Prohibition power / List dynamics	t_1-t_2	avg $z(r, i)$
Weak/Adaptive	0.3-5 0.5-10 0.5-20 0.3-30	-0.085
Medium/Adaptive	0.3-5 0.8-5 0.3-20 0.5-20 0.8-20	-0.087
Strong/Adaptive	0.3-10 0.5-10 0.5-30	0.010
Weak/Interval	5-5	-0.128
Medium/Interval	5-5 20-25 20-30	-0.084
Strong/Interval	25-25	0.068

Once good values for the parameters have been determined we study main effects and interactions of the two remaining qualitative factors, list dynamics and prohibition power. To this aim, we run a full factorial design collecting 25 replicates per instance. Each experimental unit exploits a different combination of list dynamics, prohibition power and test instance. The result $r(i)$ is the penalty

Main Effects	Pr(> F)	Interaction Effects	Pr(> F)
Prohibition power	0.0669	Prohibition p. List dyn.	0.3738
List Dynamics	0.6896		—

(b) Numerical results of ANOVA on EXAMINATION TIMETABLING

for constraint violation, which has to be minimised. In order to have comparable values across all the tested instances we consider the z-score $z(r, i) = \frac{r(i) - \bar{r}(i)}{s[r(i)]}$ as the response variable in our analysis.

In the analysis of variance instances are treated as blocks and hence their influence on the performance of the algorithms, though recognised, is not taken into account (this entails that the results of the analysis are robust with respect to the set of instances). The results of the F-ratio and the Friedman test indicate as non significant both the main effects and the interactions (The F-ratios are shown in Table 3b, whereas the Friedman test yields a p-value of 0.54).

The visual investigation of results by means of box-plots in Figure 2 reveals indeed that the numerical differences in results are apparently negligible. In conclusion, if the quantitative parameters are well chosen by means of a statistically sound procedure, all configurations of qualitative parameters are equally good.

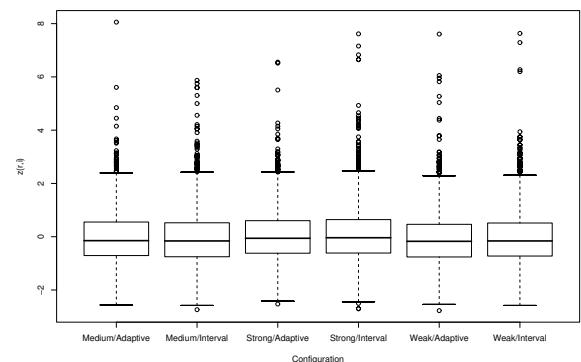


Figure 2: Results of the six configurations for the qualitative features

Finally we compare, in Table 4, our overall best results (in bold) with the currently published ones (see references on the column labels). (The use of best results to infer performance is known in Statistics to be a biased procedure, here however it is justified by the usual practise in the literature.) From Table 4 it is evident that we improved significantly w.r.t. both our previous best results [11] and other results obtained by using similar local search techniques [6]. In addition, although our results are still far from the best results attained by Merlot et al. [21], they are in most cases the second best ones.

Table 4: Best results found for each configuration and comparison with the best known results.

Instance	<i>p</i>	W/A	W/D	W/F	M/A	M/D	M/F	S/A	S/D	S/F	[11]	[5]	[6]	[21]
car-f-92	40	271	292	265	263	278	275	297	224	242	424	331	268	158
car-s-91	51	46	38	38	53	54	32	37	40	33	88	81	74	31
kfu-s-93	20	1148	1027	1103	1047	914	984	1172	1104	932	512	974	912	247
nott	23	112	89	109	103	106	69	109	112	73	123	269	—	7
nott	26	17	17	17	7	15	15	16	13	20	11	53	—	—
tre-s-92	35	0	0	0	0	0	0	0	0	0	4	3	2	0
uta-s-92	38	534	544	526	584	572	507	565	537	567	554	772	680	334

7 RELATED WORK & CONCLUSIONS

This study has been inspired by the works [2] and [19], and by the more recent work [3], which in different ways trace the guidelines for a correct empirical analysis of algorithms. The application of these guidelines to analyse TS features has been proposed in [28]. With respect to these works, we take into consideration also interaction effects between factors and hence remove the bias of a one-feature-at-a-time procedure.

The work in [27] proposes a similar analysis, although it considers specifically the *lesser* used features of TS, whereas we focus on the most used ones. In addition, also that work uses the one-feature-at-a-time approach, thus missing the interactions among them.

We adopted different statistical procedures, such as ANOVA, Friedman test and racing procedures to corroborate the results and guarantee fairness to all TS features. On the GRAPH COLOURING we have been able to discriminate which components of neighbourhood exploration and prohibition power are important to be selected correctly. Less relevant is instead the choice of the tabu length.

On the EXAMINATION TIMETABLING the analysis provides a similar indication to the results found on the GRAPH COLOURING. No significant influence of both main and interaction effects on the qualitative factors considered have been detected. Nevertheless, the analysis permits to improve the tuning of the numerical parameters and improve over previously published configurations. The results of the final algorithm are comparable with the state-of-the-art solvers.

A common conclusion arising in both studies is that, once a good setting has been found for the quantitative parameters, the differences between some of the qualitative parameters are less pronounced.

More features could be added to TS implementations. One example is the case in which the neighbourhood explored is the union of a set of basic neighbourhood relations (see [12]). In this case, typically, there are different tabu dynamics for moves of different sort. Another example, is the use of long term memory or reactive mechanisms. Extending the scope of this analysis to these and similar cases will be the subject of future work.

REFERENCES

- [1] H. Akaike, ‘A new look at statistical model identification’, *IEEE Trans. on Automatic Control*, **19**, (1974).
- [2] R. Barr, B. Golden, J. Kelly, M. Resende, and W. Stewart, ‘Designing and reporting on computational experiments with heuristic methods’, *J. of Heur.*, **1**, 9–32, (1995).
- [3] T. Bartz-Beielstein, ‘Experimental analysis of evolution strategies – overview and comprehensive introduction’, Technical Report Reihe CI 157/03, SFB 531, Universität Dortmund, Germany, (2003).
- [4] M. Birattari, *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*, Ph.D. dissertation, Université Libre de Bruxelles, Belgium, 2004.
- [5] E. Burke, J. Newall, and R. Weare, ‘A memetic algorithm for university exam timetabling’, in *Proc. of ICPTAT-95*, pp. 241–250, (1995).
- [6] M. Caramia, P. Dell’Olmo, and G. F. Italiano, ‘New algorithms for examination timetabling’, in *Proc. of WAE 2000*, eds., S. Nher and D. Wagner, vol. 1982 of *LNCS*, pp. 230–241. Springer-Verlag, (2000).
- [7] M. W. Carter. Examination timetabling dataset. URL: <ftp://ftp.mie.utoronto.ca/pub/carter/testprob/>, 2005. Viewed: January 19, 2006, Updated: June 7, 2005.
- [8] M. Chiarandini, D. Basso, and T. Stützle, ‘Statistical methods for the comparison of stochastic optimizers’, in *Proc. of MIC2005*, eds., K. F. Doerner, M. Gendreau, P. Greistorfer, W. J. Gutjahr, R. F. Hartl, and M. Reimann, Vienna, Austria, (August 2005).
- [9] W.J. Conover, *Practical Nonparametric Statistics*, John Wiley & Sons, New York, USA, third edn., 1999.
- [10] A. Dean and D. Voss, *Design and Analysis of experiments*, Springer Texts in Statistics, Springer-Verlag, New York, NY, USA, 1999.
- [11] L. Di Gaspero and A. Schaerf, ‘Tabu search techniques for examination timetabling’, in *Proc. of PATAT-2000*, eds., E. Burke and W. Erben, vol. 2079 of *LNCS*, 104–117, Springer-Verlag, (2001).
- [12] L. Di Gaspero and A. Schaerf, ‘Neighborhood portfolio approach for local search applied to timetabling problems’, *J. of Math. Modeling and Algorithms*, **5**(1), 65–89, (2006).
- [13] R. Dorne and J. Hao, ‘A new genetic local search algorithm for graph coloring’, in *Proc. of PPSN V, 5th Int. Conference*, eds., A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, vol. 1498 of *LNCS*, 745–754, Springer-Verlag, (1998).
- [14] M. R. Garey and D. S. Johnson, *Computers and Intractability—A guide to NP-completeness*, W.H. Freeman & Co., San Francisco, 1979.
- [15] Fred Glover and Manuel Laguna, *Tabu search*, Kluwer Ac. Publ., 1997.
- [16] A. Hertz and D. de Werra, ‘Using tabu search techniques for graph coloring’, *Computing*, **39**, 345–351, (1987).
- [17] J. N. Hooker, ‘Needed: An empirical science of algorithms’, *Operations Research*, **42**(2), 201–212, (1996).
- [18] H. H. Hoos and T. Stützle, *Stochastic Local Search Foundations and Applications*, Morgan Kaufmann Publishers, San Francisco, 2005.
- [19] D. S. Johnson, ‘A theoretician’s guide to the experimental analysis of algorithms’, in *Data Structures, Near Neighbor Searches, and Methodology: 5th and 6th DIMACS Impl. Challenges*, eds., M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, 215–250, AMS, (2002).
- [20] F. T. Leighton, ‘A graph coloring algorithm for large scheduling problems’, *J. Res. Natl. Bur. Standards*, **84**, 489–506, (1979).
- [21] L. Merlot, N. Boland, B. Hughes, and P. Stuckey, ‘A hybrid algorithm for the examination timetabling problem’, in *Proc. of PATAT-2002*, eds., E. Burke and P. De Causmaecker, vol. 2740 of *LNCS*, pp. 207–231. Springer-Verlag, (2003).
- [22] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, ‘Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems’, *Artificial Intelligence*, **58**, 161–205, (1992).
- [23] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, sixth edn., 2005.
- [24] B.M.E. Moret, ‘Towards a discipline of experimental algorithmics’, in *Data Structures, Near Neighbor Searches, and Methodology: 5th and 6th DIMACS Impl. Challenges*, eds., M.H. Goldwasser, Johnson D.S., and C.C. McGeoch, num. 59 in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 197–213, AMS, (2002).
- [25] R. L. Rardin and R. Uzsoy, ‘Experimental evaluation of heuristic optimization algorithms: A tutorial’, *J. of Heur.*, **7**(3), 261–304, (2001).
- [26] A. Schaerf, ‘A survey of automated timetabling’, *AI Review*, **13**(2), 87–127, (1999).
- [27] I. Whittley and G. Smith, ‘Evaluating some lesser used features of the tabu search algorithm’, in *4th EU/MEME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*, (2004).
- [28] J. Xu, S. Chiu, and F. Glover, ‘Fine-tuning a tabu search algorithm with statistical tests’, *Intl. Tran. in OR*, **5**(3), 233–244, (1998).

Random Subset Optimization

Boi Faltings and Quang-Huy Nguyen¹

Abstract. Some of the most successful algorithms for satisfiability, such as Walksat, are based on random walks. Similarly, local search algorithms for solving constraint optimization problems benefit significantly from randomization. However, well-known algorithms such as stochastic search or simulated annealing perform a less directed random walk than used in satisfiability. By making a closer analogy to the technique used in Walksat, we obtain a different kind of randomization called *random subset optimization*. Experiments on both structured and random problems show strong performance compared with other local search algorithms.

1 Introduction

Large combinatorial optimization problems often cannot be solved exactly. Approximations are obtained using *local search*: starting from an initial state, the algorithm makes a sequence of local changes, called *moves*, chosen to improve the quality of the current solution as much as possible. One of the main drawback in this strategy is that it often gets stuck in local optima, where the objective function can only be improved by making changes beyond the neighbours of the current state.

To keep such procedures from getting stuck in local optima, many such algorithms use some form of randomization. The first such process was simulated annealing ([7]), where with some probability *LocalChoice* accepts a move that degrades the objective function. These moves let the search escape from local optima and give it the possibility to restart hillclimbing from a hopefully better starting point. Other algorithms, such as stochastic local search ([2]), randomly allow moves to neighbouring states that only give some, but not necessarily the maximal improvement in the objective function. Purely random moves have the disadvantage that they are not guided towards actually breaking out of a local optimum. An exception are the algorithms for SAT problems, in particular Walksat and its variants ([11, 9]), where only moves that satisfy at least one currently unsatisfied clause are considered. This kind of random walk is more directed towards actually improving some aspect of the solution.

A direct analog to the Walksat strategy for combinatorial optimization would be to randomly choose a part of the optimization function, and make a move that is optimal with respect to this part only. It turns out that this results in a more focussed optimization strategy while retaining the power to escape from local minima. In this paper, we explore this strategy, which we call random subset optimization, in more detail.

¹ Artificial Intelligence Laboratory (LIA), Swiss Federal Institute of Technology (EPFL), email: boi.faltings—quanghuy.nguyen@epfl.ch

2 Definitions

Definition 1 A discrete constraint optimization problem (*COP*) is a tuple $\langle X, D, C, R \rangle$ where:

- $X = \{x_1, \dots, x_n\}$ is a set of variables.
- $D = \{d_1, \dots, d_n\}$ is a set of domains of the variables, each given as a finite set of possible values.
- $C = \{c_1, \dots, c_p\}$ is a set of hard constraints, where c_i is a function $d_{i1} \times \dots \times d_{il} \rightarrow \{0, 1\}$ that returns 1 if the value combination is allowed and 0 if it is not.
- $R = \{r_1, \dots, r_o\}$ is a set of soft constraints, where r_i is a function $d_{i1} \times \dots \times d_{il} \rightarrow \mathbb{R}$ giving the weight of each combination of values.

A solution to a COP is an assignment of values to all variables that satisfies all hard constraints and maximizes (or minimizes) the sum of the soft constraints. We also call the sum of soft constraints the *objective function* of the COP.

Many optimization problems can be formulated as a COP. For example, soft graph coloring can be formulated as a COP where vertices are represented by variables, domains are sets of available colors, and soft constraints correspond to weighted edges of the graph. Another direct example is the weighted Max-SAT problem where the goal is to minimize the sum of unsatisfied clause weights. These problems have all been shown to be NP-complete or NP-hard [4].

The local search framework we assume in this paper is given in Algorithm 1.

Algorithm 1 Local search algorithm for COP

```

procedure LocalSearch( $X, D, C, R$ )
     $\underline{v} \leftarrow SelectInitialSolution(X, D, C, R)$ 
    repeat
         $\underline{v}^{old} \leftarrow \underline{v}$ 
         $x_k \leftarrow ChooseRandom(X)$ 
         $N \leftarrow ChooseNeighbours(x_k, \underline{v}^{old}, X, D, C, R)$ 
         $\underline{v} \leftarrow LocalChoice(N, R)$ 
    until termination condition met
    return  $\underline{v}$ 
end procedure

```

The algorithm starts from a complete assignment of values to all variables, and then proceeds iteratively by local improvements. The initial assignment can be chosen randomly, but must be consistent with the hard constraints. If such a solution is too difficult to find, it may be useful to change some of the hard constraints to soft constraints with high penalty.

Search is governed by an iterative application of two functions. First, function *ChooseNeighbours* provides a set of candidate assignments that are close to the current one and could possibly improve

it. In our implementation, they assignment that are equal to \underline{v} except that they assign a randomly selected variable x_k all other values in its domain that are consistent with the rest of \underline{v} according to the hard constraints. Second, the assignment \underline{v} is updated using the function *LocalChoice*. It chooses a new assignment to optimize the sum of soft constraints in R .

The iteration continues until a termination condition is met, for example when there is no further improvement for some number of steps. To avoid getting stuck in local optima, the performance of a local search procedure is significantly improved by randomization. This means that occasionally *LocalChoice* chooses a value that may even decrease the objective function ([7, 11]).

3 Algorithms

We propose a new algorithm called *random subset optimization* (RSO). In this algorithm, the local choice is made by changing a randomly selected variable x_k . Whenever an improvement of the entire objective function is possible, i.e. the algorithm has not reached a local optimum, it chooses the value for x_k that maximizes the entire objective function R . When the algorithm has reached a local optimum, with some probability p it randomly chooses a subset of the objective function and finds a new value for x_k that only optimizes this part. In this way, the search avoids moves that only degrade the solution and rapidly lead back to the original optimum, and focusses on moves that actually have a chance to break out of it. This technique can be applied whenever the objective function can be decomposed into parts, as for example when it is given as a set of soft constraints.

Algorithm 2 Random subset local search algorithm for COP

```

procedure RSO(X,D,C,R,p,d)
1:  $\underline{v} \leftarrow SelectInitialSolution(X, D, C)$ 
2: repeat
3:    $\underline{v}^{old} \leftarrow \underline{v}$ 
4:    $x_k \leftarrow ChooseRandom(X)$ 
5:    $N \leftarrow ChooseNeighbours(x_k, \underline{v}^{old}, X, D, C, R)$ 
6:   if LocalOptimum( $\underline{v}$ ) then
7:     if random(0..1)  $\leq p$  then
8:        $R \leftarrow ChooseRandomSet(R, d)$ 
9:        $\underline{v} \leftarrow LocalChoice(N, \tilde{R})$ 
10:    else
11:       $\underline{v} \leftarrow ChooseRandom(N)$ 
12:    end if
13:   else
14:      $\underline{v} \leftarrow LocalChoice(N, R)$ 
15:   end if
16: until termination condition met
17: return  $\underline{v}$ 
end procedure

```

Algorithm RSO, shown as Algorithm 2, is a development of Algorithm 1 and introduces two random elements. First, the neighbourhood N is generated by randomly choosing a variable x_k and considering as neighbours all assignments that differ in the value given to that variable. Second, when the algorithm reaches a local optimum, the algorithm chooses with probability p to optimize only for a subset \tilde{R} of the constraints making up the objective function, and with probability $(1 - p)$ to choose a random neighbour regardless of whether it improves the objective function or not. The subset \tilde{R} is chosen by function *ChooseRandomSet* that takes as an additional parameter

d the number of soft constraints that are dropped from the set R . The algorithm terminates when a termination condition is met, for example, when there is no improvement after a certain number of steps, or when a maximum number of steps is reached.

In the experiments we report later, it appears that the best performance is reached when d is small and p is close to 1.

In another variant, called *adaptive* random subset optimization (ARSO), we let the parameter d start with a high value and decrease over time, similarly to the cooling schedule used in simulated annealing. At each value of d , the algorithm performs a linear number of LocalChoice steps w.r.t the number of variables. In our implementation, d is simply reduced by 1 after each iteration. When d decreases below 0, the algorithm terminates.

4 Completeness

The outcome of randomized algorithms cannot be predicted with certainty, and so the notion of completeness cannot be applied directly here. However, Hoos [5] has defined the notion of *probabilistically approximately complete*:

Definition 2 A randomized search algorithm is probabilistically approximately complete (PAC) if for all soluble problem instances, the probability $p(t)$ that it finds the optimal solution in time less than t goes to 1 as t goes to infinity.

Pure random walk strategies, such as picking random values without regard for whether they improve or degrade solution quality, make a local search algorithm PAC: starting with any initial state, there is a sequence of moves that sets each variable x_i to the value it has in an optimal assignment v_i^* , and this sequence of moves has a non-zero probability.

The RSO strategy by itself, reached when the parameter p of Algorithm 2 is set to 1, is clearly not PAC, as there are problems where the optimal assignment \mathbf{v}^* is not reachable by the algorithm at all. This is the case when the optimal assignment is a compromise among all relations, i.e. there is no subset of the relations for which \mathbf{v}^* is also optimal. However, when $p < 1$, the algorithm also performs a random walk with some probability and does become PAC.

5 Experimental results

We evaluated the performance of our algorithms both on structured and unstructured problems. As a structured problem, we chose network resource allocation. As unstructured problems, we chose soft k-coloring and graph coloring optimization on random graphs.

5.1 Network resource allocation problem

The network resource allocation problem consists of allocating tracks in the train network shown in Figure 1 to different operators. To avoid collisions, each arc in the graph can only be allocated to one operator who can then run trains on it. Operators transport goods between nodes in the network. Each transportation task can take one of 3 feasible routes and generates one variable whose domain is the agent and route assigned to it. For example, if task 3 (*London, Manchester*) is assigned to agent a_1 on the route (*London* \rightarrow *Birmingham* \rightarrow *Manchester*), the corresponding variable x_3 is assigned the value ($a_1, \text{London} \rightarrow \text{Birmingham} \rightarrow \text{Manchester}$). The problem is representative of a large class of resource allocation problems, for example in communication networks or in chemical plants.

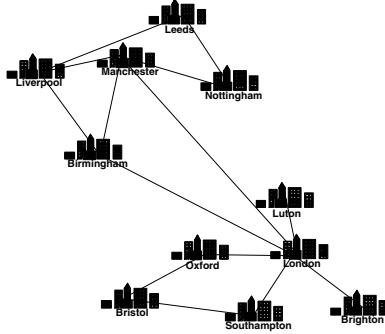


Figure 1. Network of transputers and the structure of individual processes

The network capacity is modelled by binary constraints between any pair of tasks whose routes share at least one arc. The constraint rules out assigning it to such overlapping routes but different agents. Each operator has a different and randomly generated profit margin for being assigned a demand/route combination, and declares these through its relations. These relations are modelled as soft constraints, and the objective is to maximize their sum.

We randomly generated tasks and routes and simulated the problem starting from a situation where no task is assigned to any agent. The number of agents is 30. A set of 100 tasks is generated randomly; for each task, we generate up to 3 different paths for carrying it. We always report the average performance over 100 runs.

In the first experiment, we want to see how the level of randomness affects the performance of the RSO algorithms. Figure 2 shows the average total profit obtained as we varied parameter p from 0 to 1 and kept the number of relations to be left out d always at 1. The shape of this curve does not seem to depend significantly on the value of d . It clearly shows that the performance is best with p close to 1. Thus, in the following experiments, we always set p to 1.

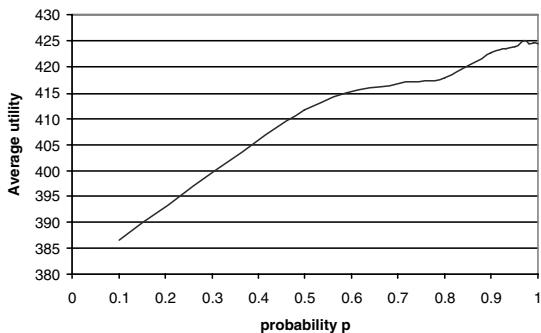


Figure 2. Variation of the average cost of the random subset optimization algorithm on network resource allocation problem, for different values of p .

Next, we are interested in what the best value of parameter d would be for the network resource allocation problem. We run the RSO algorithms with the parameter p set to 1 and with different number of relations to leave out at a local minimum. Figure 3 shows the average solution quality reached in 100 runs. The results show that the best results are obtained for small d . Thus in the following experiments, we always set the parameter p to 1 and d to 2.

To determine whether the RSO algorithms would be efficient for the network resource allocation problems, we compare the perfor-

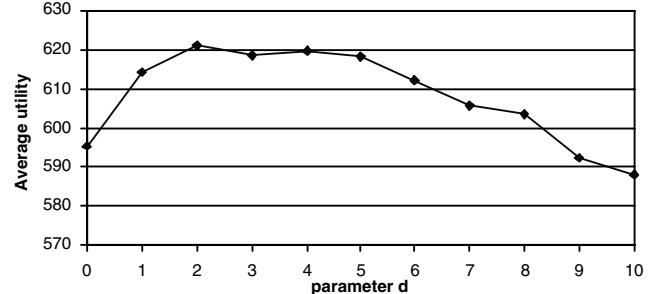


Figure 3. Variation of the average cost of the random subset optimization algorithm on network resource allocation problem, for different values of d .

mance of two versions of the RSO algorithms, RSO for Algorithm 2 with $p = 1, d = 2$ and ARSO for Adaptive RSO, with the following known local search techniques:

- HC: hill-climbing without randomization, included as a baseline comparison. To achieve better results for hill-climbing algorithm, we allow equal moves in the hill-climbing process: the algorithm stops when there is no further improvement for several iterations.
- RHC: hill-climbing with random restarts, similar to the GSAT algorithm [10]. For a fair comparison, we set the parameters so that the total number of iterations is similar to the other algorithms. The hill-climbing stops when there is no improvement for several iterations or the number of iterations exceeds 5 times of the number of variables and is then followed by a restart. The total number of iterations is limited to 2000 as for the other algorithms.
- SA: simulated annealing [7]. The temperature is initialized such that the acceptance rate at the beginning is 97%, and updated according to a geometric cooling schedule with $\alpha = 0.97$; for each temperature value, $n(n - 1)$ search steps are performed, where n is the number of variables. The search is terminated when there is no improvement of the evaluation function for five consecutive temperature values, and the acceptance rate of new solutions falls below 1%.
- SLS: stochastic local search [3, 12]. We tested different versions of the SLS algorithm on the network resource allocation problem and took the best one (corresponding to DSA-B in [12]) with the probability p set to 0.3.

All the algorithms start from an empty assignment, i.e., all the tasks are unassigned. In Figure 4, we show the gain in utility as a function of the number of steps. We observe that ARSO clearly outperforms the other algorithms, while the SA algorithm is slightly better than RSO. However, RSO converges much faster than all other algorithms, and seems to be a clear winner for this application.

5.2 Soft k-coloring

In the next experiment, we considered the soft graph coloring problem also used as a benchmark in [3, 12]. A soft k -coloring problem consists of a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. Each vertex of the graph can be colored by one of k different colors. Each edge $(u, v) \in E$ is associated with a real valued weight $w(u, v)$. An edge (u, v) is violated if u and v have the same color. Given a number k , the goal is to minimize the sum of the weights of violated edges.

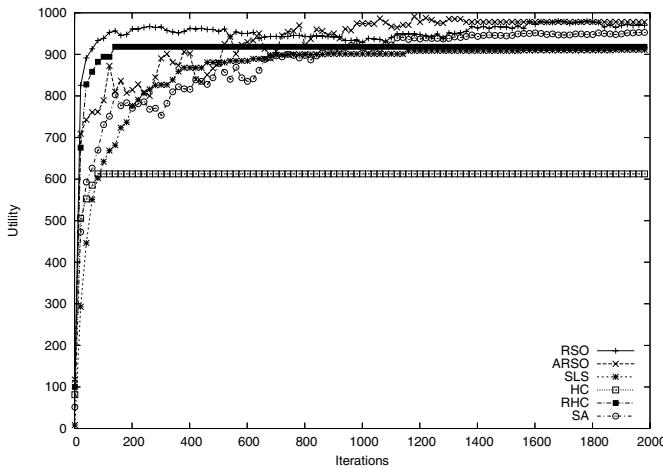


Figure 4. Average utility gain of different local search algorithms on the networks resource allocation problem as a function of the number of steps.

This problem can be modeled as a COP where each vertex is represented as a variable whose domain is the set of available colors. The set of hard constraints is empty. For every edge E in the problem, there is a soft constraint between the variables corresponding to its vertices u and v . It has value $w(u, v)$ if the colors assigned to the corresponding variables are different, and 0 otherwise.

We ran experiments on randomly generated k-coloring problems with 100 vertices and the degree of connectivity is $\approx k + 1$ where k is the number of colors used. We let the algorithms run for a large number of steps until either the overall quality does not improve for $10*n$ steps, or a timeout limit of l steps is reached, where l is variable to indicate the performance of the algorithms. We always report the average performance over 100 runs.

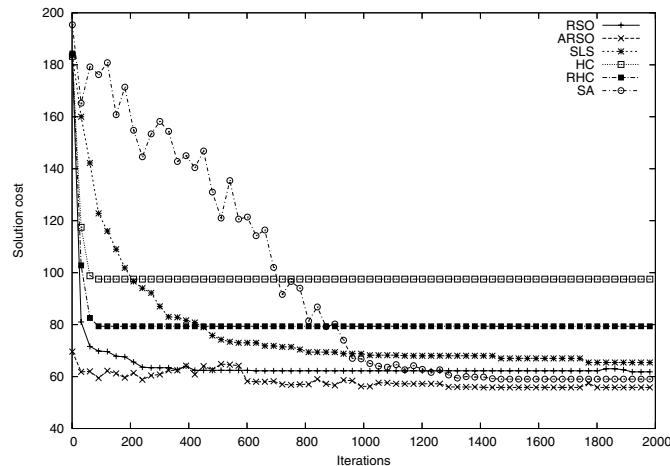


Figure 5. Average solution cost of various optimization algorithms on the soft-k-coloring problem as a function of the number of steps.

Figure 5 shows the results after 2000 iterations. We can observe that SA and RSO eventually reach about the same solution quality, and again ARSO is significantly better than the others. However, RSO converges more than 10 times faster than the SLS and simulated annealing algorithms.

5.3 Coloring graphs with a minimum number of colors

A third set of experiments was carried out on a non-decomposable problem, namely coloring graphs with a minimum number of colors. It involves an unweighted graph $G = (V, E)$ where V is the set of vertexes and E the set of edges. The goal is to find a minimum number of colors to color the vertexes of the graph so that no two vertexes connected by an edge have the same color.

This problem can be modeled as a COP where each vertex is represented as a variable whose domain is the set of available colors, and there is an extra variable x_c that represents a bound on the number of colors used. The hard constraints are that adjacent vertexes must have different colors. There are two types of soft constraints:

- Soft constraint on x_c : the smaller value the better ($cost = x_c$)
- Soft constraints on all variables: the color has to be less than x_c or else there is a penalty of ($cost = 2 * (color - x_c)$). This forces colors to be lower than the maximum number, and is modelled as a soft constraint so that local moves that change only one variable can easily move between consistent assignments.

The initial state is assigned randomly such that all neighbouring nodes have different colors, for example by choosing a different color for every node. The modifications of the variables during the search are chosen so as to continuously satisfy these hard constraints. Note that this is a minimization problem.

We ran the same local search algorithms on random hard 3-coloring problem instances with 100 vertices generated by the procedure of [1]. We started with 10 colors ($x_c = 10$) and set the initial assignment of the COP such that all the vertices have different colors. Figure 6 compares the performance of random subset optimization (RSO) and adaptive random subset optimization (ARSO) with the same algorithms as for the graph coloring problem (HC-hillclimbing without randomization, RHC-hillclimbing with random restarts, SLS-stochastic local search, and SA-simulated annealing). Figure 7 shows the average number of colors used in different local search algorithms.

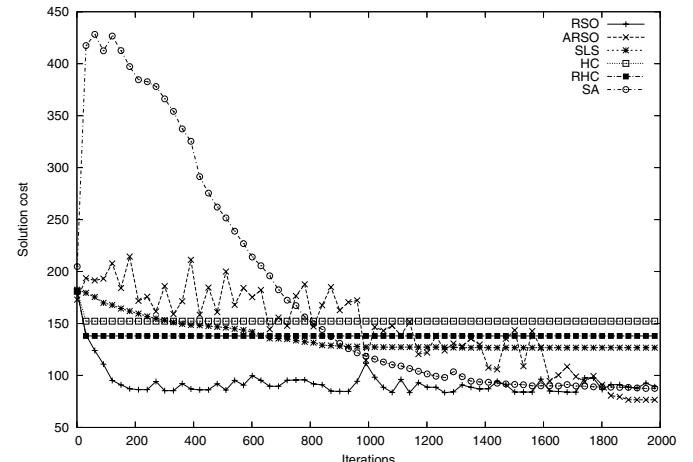


Figure 6. Average solution cost of different local search algorithms on the min-coloring problem as a function of the number of steps.

We can see that among the earlier local search algorithms, only the ARSO algorithm achieves a significant optimization of the number of colors used and thus outperform all other algorithms. RSO and

SA have similar performance. However, RSO converges much faster than the others. This fact was surprising to us, given that we cannot guarantee any completeness result on such a non-decomposable problem.

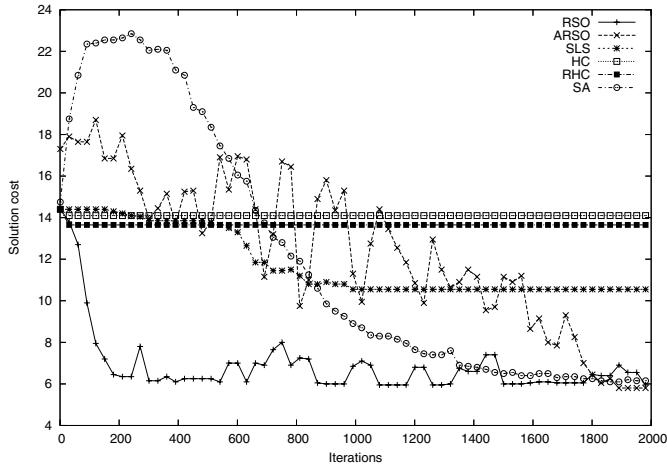


Figure 7. Average number of colors used of different local search algorithms as a function of the number of steps.

One aspect that may appear strange is that in simulated annealing (SA) and adaptive random subset optimization (ARSO), the number of colors appears to be very low in the beginning and only increases gradually. This is explained when considering the evolution of the cost of constraint violations shown in Figure 6. Both algorithms are very tolerant of soft constraint violations, and thus can start with a small limit on the number of colors that is often violated. As the number of constraint violations decreases, both algorithms use more and more colors. Only towards the end does the number of colors decrease while also satisfying the coloring constraints. We also see that the high degree of randomization in these algorithms makes their behavior very unstable initially, so that care must be taken to allow sufficient time for convergence.

5.4 Comparison of computation times

Table 1 shows the running time of the algorithms on the three benchmark problems, using an Intel PC with a 2GHZ clock rate. For RSO, ARSO, HC, RHC, and SLS, the computation times to complete one iteration are similar as they all perform a single local move. Hill-climbing (HC and RHC) often terminate earlier when they reach a local optimum so have shorter run times. The SA algorithm takes significantly longer as it makes many moves at for each temperature value.

Problem	RSO	ARSO	SLS	HC	RHC	SA
NRA	89.01	113.45	96.23	3.53	7.87	1905.73
soft k-coloring	21.25	26.74	14.19	1.06	2.84	352.5
Min-coloring	19.71	25.37	12.43	0.97	2.63	351.32

Table 1. Running times in seconds of different algorithms, for convergence or 2000 iterations.

6 Conclusions

Local search algorithms are often the only possibility to solve large optimization problems. Randomization has been shown to be key to obtaining results that are close to the optimal solution. This strategy has been particularly successful for SAT problems.

For general optimization problems, simulated annealing often achieves the best results, but has the drawback that it converges very slowly. By drawing an analogy with techniques that have been successful for SAT problems, we developed the technique of random subset optimization that empirically converges much faster than known algorithms while obtaining solutions of equal or better quality.

We are now conducting further experiments to determine the influence of different parameters and variations on the performance of the algorithms.

Acknowledgement

This work was supported by the Swiss National Science Foundation under contract No. 200020-103421.

REFERENCES

- [1] H. M. Adorf and M. D. Johnston, ‘A discrete stochastic neural network algorithm for constraint satisfaction problems’, in *In Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, (1990).
- [2] M. Fabiunke, ‘Parallel distributed constraint satisfaction’, in *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 99)*, pp. 1585–1591, Las Vegas, (1999).
- [3] Stephen Kirkpatrick and Lambert Meertens, ‘An experimental assessment of a stochastic, anytime, decentralized, soft colourer for sparse graphs’, in *First Symposium on Stochastic Algorithms: Foundations and Applications*, ed., Kathleen Steinhoefel, 49–64, Springer-Verlag, (2001).
- [4] Michael R Garey and David S Johnson, *Computers and Intractability*, W. H. Freeman and Company, 1979.
- [5] Holger H. Hoos, ‘On the run-time behaviour of stochastic local search algorithms for sat’, in *Proceedings of the AAAI ’99/IAAI ’99*, pp. 661–666. AAAI, (1999).
- [6] Holger H. Hoos and Thomas Sttzle, *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann, San Francisco (CA), USA, 2004.
- [7] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, ‘Optimization by simulated annealing’, *Science*, **220**(4598), 671–680, (1983).
- [8] Roger Mailler and Victor Lesser, ‘Solving distributed constraint optimization problems using cooperative mediation’, in *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pp. 438–445, (2004).
- [9] D. McAllester, B. Selman, and H. Kautz, ‘Evidence for invariants in local search’, in *In Proceedings of AAAI-97*, pp. 321–326, (1997).
- [10] B. Selman, H. Levesque, and D. Mitchell, ‘A new method for solving hard satisfiability problems’, in *Proceedings of the AAAI-92*, pp. 440–446, (1992).
- [11] B. Selman, H. Levesque, and D. Mitchell, ‘Noise strategies for improving local search’, in *Proceedings of the AAAI-94*, pp. 337–343, (1994).
- [12] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg, ‘Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks’, *Artificial Intelligence*, **1-2**(161), 55–88, (2005).

Search for Compromise Solutions in Multiobjective State Space Graphs

Lucie Galand and Patrice Perny¹

Abstract. The aim of this paper is to introduce and solve new search problems in multiobjective state space graphs. Although most of the studies concentrate on the determination of the entire set of Pareto optimal solution paths, the size of which can be, in worst case, exponential in the number of nodes, we consider here more specialized problems where the search is focused on Pareto solutions achieving a well-balanced compromise between the conflicting objectives. After introducing a formal definition of the compromise search problem, we discuss computational issues and the complexity of the problem. Then, we introduce two algorithms to find the best compromise solution-paths in a state space graph. Finally, we report various numerical tests showing that, as far as compromise search is concerned, both algorithms are very efficient (compared to MOA*) but they present contrasted advantages discussed in the conclusion.

1 INTRODUCTION

Traditionally, heuristic search problems in state space graphs are considered in the framework of single objective optimization. The value of a path between two nodes is defined as the sum of the costs of its arcs and the problem amounts to finding one path with minimum cost among all paths from a source node to the goal. This problem is solved by classical constructive search algorithms like A^* [5] which provide, under certain assumptions, the optimal solution-path, using a heuristic function to concentrate the search on the most promising nodes and sub-paths. Thus, during the search, preferences over sub-paths or nodes candidate to development are measured by a scalar evaluation function inducing a complete weak-order over competing feasible paths.

However preferences are not always representable by a single criterion function. Indeed, in many practical search problems considered in Artificial Intelligence (e.g. path planning, game search, preference-based configuration), comparison of solutions includes several aspects or points of view (agents, criteria, scenarios). In such problems, an action allowing a transition from a node to another is evaluated with respect to these different points of view, thus leading to multiple conflicting evaluations non-necessarily reducible to a single overall cost. For this reason, the vector-valued generalization of the initial problem have been investigated and several extensions of the so-called A^* algorithm are now available. Let us mention, among others, MOA* the multiobjective extension of A^* finding all Pareto optimal paths [13] and its recent refinement [9], U* a variation of A^* used to find a maximal path with respect to a multiattribute utility function [2], ABC* to find paths which best satisfy a set prioritized goals [8] and finally preference-based generalizations of MOA* [6] and [10].

As soon as several objectives must be optimized simultaneously, most of the studies concentrate on the determination of the entire set of Pareto optimal solutions (here solution-paths having cost vector that cannot be improved on a criterion without being degraded on another criterion). However, the size of the Pareto set in combinatorial problems being often very large, its determination induces prohibitive response times and requires very important memory space.

Nevertheless, in many contexts, there is no need to determine the entire set of Pareto optimal elements, but only specific compromise solutions within this set. This is particularly the case in multiobjective path-planning problems where the most preferred solution-paths are those achieving a good compromise between various conflicting objective (minimize time, energy consumption, risk, . . .). This is also the case in vector-valued path problems issued from robust optimization. In such problems involving a single cost function, several plausible scenarios are considered, each having a different impact on the costs of transitions between nodes. The aim is to find a *robust path*, that is a path which remains suitable in all scenarios (e.g. the minimax path [7, 11]). In such examples, the type of compromise solution sought in the Pareto set is precisely defined and this can be used to speed up the search.

When preference information is not sufficient to formulate a stable overall preference model, iterative compromise search can still be used to explore the Pareto set. One starts with a “neutral” initial preference model used to generate a well-balanced compromise solution within the Pareto set and the model progressively evolves with feedbacks during the exploration to better meet decision maker’s desiderata. Such an interactive process is used in multiobjective programming on continuous domains to scan the Pareto set which is infinite, see e.g. [12]. The same approach is worth investigating in large size combinatorial problems when complete enumeration of the Pareto set is not feasible.

Motivated by such examples, we consider in this paper multiobjective search problems in state space graphs with a specific focus on Pareto solution-paths achieving a good compromise between the conflicting objectives. After introducing the basic notations and definitions used in the paper, we define formally the compromise search problem in Section 2 and discuss computational issues. In section 3 we recall the basic principles of MOA* and introduce two original algorithms to find the best compromise solution-paths in a state space graph. The first one, called BCA* (Best Compromise A^*), is a variant of MOA* specially designed to focus directly on a target compromise solution. The second one called kA* achieves the same goal while working on a scalarized version of the problem. Section 4 is devoted to the presentation of numerical tests allowing the relative performance of MOA*, BCA* and kA* to be compared, in the determination of the best compromise solution-paths.

¹ LIP6 – University of Paris 6, France, email: firstname.name@lip6.fr

2 COMPROMISE SEARCH

We introduce now more formally the framework of multiobjective optimization and compromise search in state space graphs.

2.1 Notations and Definitions

A^* algorithm and its multiobjective extensions look for “optimal” paths (in some sense) in a state space graph $G = (N, A)$ where N is a finite set of nodes (possible states), and A is a set of arcs representing feasible transitions between nodes. Formally, we have $A = \{(n, S(n)), n \in N\}$ where $S(n) \subseteq N$ is the set of all successors of node n (nodes that can be reached from n by a feasible elementary transition). Then $s \in N$ denotes the source of the graph (the initial state), $\Gamma \subseteq N$ the subset of goal nodes and $\mathcal{P}(s, \Gamma)$ denotes the set of all paths from s to a goal node $\gamma \in \Gamma$, and $\mathcal{P}(n, n')$ the set of all paths linking n to n' . We call *solution-path* a path from s to a goal node $\gamma \in \Gamma$. Throughout the paper, we assume that there exists at least one finite length solution path.

We consider a finite number of criteria $v_i : A \rightarrow \mathbb{N}$, $i \in Q = \{1, \dots, q\}$. We assume that v_i 's are cost functions to be minimized. Hence, state space graph G is endowed with a vector valuation function $v : A \rightarrow \mathbb{N}^q$ which assigns to each arc $a \in A$ the cost vector $v(a) = (v_1(a), \dots, v_q(a))$.

For any path P , the vector $p = v(P) = \sum_{a \in P} v(a)$ gives the costs of P with respect to criteria v_i , $i \in Q$ (costs are supposed to be additive). In other words, p is the image of P in the space of criteria \mathbb{N}^q . For any asymmetric dominance relation \succ defined on a set $X \subseteq \mathbb{N}^q$, the set $\{x \in X : \nexists y \in X, y \succ x\}$ is said to be the set of *non-dominated elements in X with respect to \succ* . It is non-empty as soon as \succ is acyclic. We recall now the basic dominance relations and related non-dominated sets used in multiobjective optimization:

Definition 1 The Pareto dominance relation (*P-dominance* for short) on cost-vectors of \mathbb{N}^q is defined by:

$$x \succ y \iff [\forall i \in Q, x_i \leq y_i \text{ and } \exists i \in Q, x_i < y_i]$$

Within a set $X \subseteq \mathbb{N}^q$ any element x is said to be dominated when $y \succ x$ for some y in X , and non-dominated when there is no y in X such that $y \succ x$.

Definition 2 The strict Pareto dominance relation (*SP-dominance* for short) on cost-vectors of \mathbb{N}^q is defined by:

$$x \succ y \iff \forall i \in Q, x_i < y_i$$

Within a set $X \subseteq \mathbb{N}^q$ any element x is said to be strictly dominated when $y \succ x$ for some y in X , and weakly non-dominated when there is no y in X such that $y \succ x$.

Within X the set of P-non-dominated elements (also called the Pareto set) is denoted $ND(X)$ whereas the set of weakly P-non-dominated elements is denoted $WND(X)$. Note that $ND(X) \subseteq WND(X)$ since $x \succ y \Rightarrow x \succ y$ for all $x, y \in \mathbb{N}^q$. For any set $X \in \mathbb{N}^q$, the Pareto set $ND(X)$ is bounded by the ideal point α and the nadir point β with components $\alpha_i = \inf_{x \in ND(X)} x_i = \inf_{x \in X} x_i$ and $\beta_i = \sup_{x \in ND(X)} x_i$, for all $i \in Q$. The multiobjective optimization problem can be stated as follows:

Multiobjective Search Problem. We want to determine the entire set $ND(\{v(P), P \in \mathcal{P}(s, \Gamma)\})$ and the associated solution paths.

This problem is solved by MOA* (see [13]), a vector-valued version of A^* designed to determine the Pareto set in multiobjective problems. Unfortunately, the computation of the whole Pareto set

can be prohibitive in practice due to the size of the Pareto set which grows, in worst case, exponentially with the number of nodes. This can easily be shown by considering the family of instances introduced in [4]:

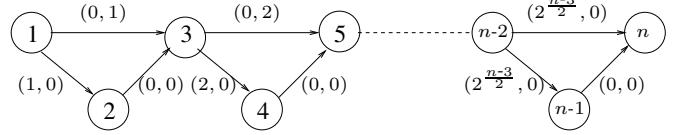


Figure 1. Hansen Graph

In the graph of Figure 1, we have an odd number of nodes n and $2^{\frac{n-1}{2}}$ distinct paths from node 1 to node n , leading to $2^{\frac{n-1}{2}}$ distinct points $(v_1(k), v_2(k)) = (k, 2^{\frac{n-1}{2}} - 1 - k)$, $k \in \{0, \dots, 2^{\frac{n-1}{2}} - 1\}$ in the space of criteria. By construction, these points are on the same line (they all satisfy $v_1(k) + v_2(k) = 2^{\frac{n-1}{2}} - 1$) orthogonal to vector $(1, 1)$; therefore, they are all Pareto optimal. Although pathological, such instances show that the determination of the Pareto set might reveal intractable in practice on large size instances. Numerical tests presented in Section 4 will confirm this point.

Moreover, even when the size of Pareto set is “reasonable”, the great majority of its elements do not provide interesting compromise solutions in practice. For instance, as soon as the criteria are conflicting, which is frequent in practice (e.g. cost vs quality, efficiency vs energy consumption), a path reaching the optimal value with respect to one criterion is often bad with respect to the other criteria. Such a path would not be a good compromise solution, yet it is a Pareto optimal solution. For this reason, it is worth defining more precisely the type of compromise sought in the Pareto set, so as then to focus the search on that type of compromise solution.

2.2 Looking for compromise solutions

A natural attitude to characterize good compromise solutions within the Pareto set is to define a scalarizing function: $s : \mathbb{N}^q \rightarrow \mathbb{R}$ in such a way that the comparison of two paths P and P' represented by performance vectors p and p' reduces to the comparison of scalars $s(p)$ and $s'(p)$. In this respect, one might think that resorting to linear scalarization of type: $s_1(p, \omega) = \sum_{i \in Q} \omega_i p_i$ where ω is a strictly positive weighting vector, is appropriate. Indeed, on the one hand, the path minimizing $s_1(p, \omega)$ over $\mathcal{P}(s, \Gamma)$ can easily be obtained. It is sufficient to scalarize the problem by considering the valuation $w : A \rightarrow \mathbb{R}$ derived from v by setting $w(a) = s_1(v(a), \omega)$ and to apply A^* with valuation w . On the other hand, weighted sum is often seen as a natural aggregator to identify good compromise solutions. Although the first argument (procedural) is perfectly correct, the second is not valid on a non-convex set (which is the case here). The point is that minimizing s_1 over the set of solution paths can only reach *supported solutions*, i.e. those P-non-dominated solutions lying on the boundary of the convex hull of the points representing feasible solution-paths in the space of criteria. Hence there might exist feasible paths achieving a very well-balanced compromise solution that cannot be obtained by optimizing $s_1(p, \omega)$, whatever the choice of the weighting vector ω . This problem, well known in multiobjective optimization, is illustrated by Example 1.

Example 1 Consider the bicriteria problem illustrated in Figure 2. In this graph, we have 4 paths from 1 (source node) to 4 (goal node): $P_1 = \langle 1, 2, 4 \rangle$, $P_2 = \langle 1, 2, 3, 4 \rangle$, $P_3 = \langle 1, 3, 4 \rangle$ and $P_4 = \langle 1, 3, 2, 4 \rangle$ with $v(p_1) = (20, 2)$, $v(p_2) = (12, 11)$, $v(p_3) = (3, 19)$ and $v(p_4) = (16, 15)$.

The graph to the right in Figure 2 represents the image of these solution-paths in the space of criteria. However we have $s_1(p_1, \omega) = 20\omega_1 + 2\omega_2$, $s_1(p_2, \omega) = 12\omega_1 + 11\omega_2$, $s_1(p_3, \omega) = 3\omega_1 + 19\omega_2$ and $s_1(p_4, \omega) = 16\omega_1 + 15\omega_2$. Since s_1 is to be minimized, P_1 is optimal when $\omega_1 < \omega_2$, P_3 is the optimal solution-path when $\omega_2 < \omega_1$ and both P_1 and P_3 are optimal when $\omega_1 = \omega_2$. We can notice that P_2 and P_4 cannot be optimal, whatever the choice of ω_1 and ω_2 . Although this is quite justified for P_4 which is P -dominated by P_2 this is problematic for P_2 which is clearly the best compromise solution path in this example. This is due to the fact that P_2 lies inside the convex hull of solutions (the grey triangle) and not on the boundary.

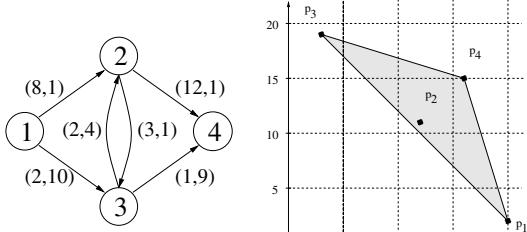


Figure 2. A bicriteria problem and its representation in the space of criteria

A better way of defining the notion of compromise solution for a solution-path P is to consider the distance between $p = v(P)$ and a prescribed reference vector r (representing ideal performances, or DM's aspirations levels). This leads to formulate the following scalarizing function:

$$s(p, r, \omega) = \| \omega \cdot (p - r) \|_{\infty} = \max_{i \in Q} \{ \omega_i |p_i - r_i| \}$$

in which ω is a weighting vector. The choice of Tchebycheff norm focus on the worst component and therefore guarantees that only feasible solutions close to reference point r on every component will receive a good score. This promotes well-balanced solutions. It also implies that iso-preference curves of type $s(x, r, \omega) = k$, for fixed r, ω and k , perfectly match P -dominance cones, which implies two important properties on any subset $X \subseteq \mathbb{N}^q$ (see [15]):

P1. If $\forall i \in Q, \omega_i > 0$, then all solutions x minimizing $s(x, r, \omega)$ over the set X of feasible cost vectors belongs to $WND(X)$. Moreover at least one of these solutions belongs to $ND(X)$.

P2. If $r > \alpha$ then for all $x \in ND(X)$ there exist strictly positive weights $\omega_i, i \in Q$ such that x is the unique solution minimizing $s(x, r, \omega)$ over X .

Property P1 shows that minimizing s yields well-balanced Pareto optima located in the set $ND(X) \cap S(X) \neq \emptyset$ where $S(X) = \arg \min_{x \in X} s(x, r, \omega)$. Property P2 shows that any type of compromise solution can be obtained within the Pareto set, with the appropriate choice of parameters r and ω . This second property explains why s , as a scalarizing function, is preferred to s_1 in multiobjective optimization on non-convex sets [15, 12].

We define normalization weights as follows $\omega_i = \delta_i / (\beta_i - \alpha_i)$, for all $i \in Q$ where $[\alpha_i, \beta_i]$ represents the range of possible values for criterion i in the subarea of interest. Optimizing s generates a weakly non-dominated path as close as possible to reference point r for the weighted Tchebycheff metric. Weights used in the metric can possibly be modified via coefficients δ_i so as to give more importance to some criteria. Function s induces a new dominance relation \succ_s on \mathbb{N}^q , defined by $x \succ_s y \Leftrightarrow s(x, r, \omega) < s(y, r, \omega)$ which means that x is a better compromise than y .

As mentionned in the introduction, compromise search can be seen as an interactive process. Initial parameters of function s are initially chosen so as to focus on the center of the Pareto set but are dedicated

to evolve during the interactive exploration using standard updating techniques [12]. To save space, we now concentrate on the main technical difficulty: the fast generation on the s -optimal path at a given step. This leads to the compromise search problem.

2.3 The compromise search problem

Compromise Search Problem. We want to determine all P -non-dominated s -optimal feasible cost-vectors (i.e. $ND(X) \cap S(X)$ with $X = \{v(P), P \in \mathcal{P}(s, \Gamma)\}$) and the associated solution-paths.

Coming back to Example 1, with $r = (0, 0)$ and $\omega = (1, 1)$ we get: $s(p_1, r, \omega) = 20$, $s(p_2, r, \omega) = 12$, $s(p_3, r, \omega) = 19$ and $s(p_4, r, \omega) = 16$. Hence the preferred compromise according to \succ_s is P_2 with cost-vector $p_2 = (12, 11)$ which seems quite natural here. Moreover, sub-paths from node 1 to node 3 are P with cost $p = (11, 2)$ and P' with $p' = (2, 10)$. Since $p' \succ_s p$ the locally best compromise-path at node 3 is P' . However, appending path $P'' = \langle 3, 4 \rangle$ with cost $p'' = (1, 9)$ to P and P' respectively yields $p + p'' = (12, 11)$ and $p' + p'' = (3, 19)$, so that $p' + p'' \prec_s p + p''$, a preference reversal. Notice that the optimal path $P_2 = P \cup P''$ would be lost during the search if P is pruned at node 3 due to P' . Although the dominance relation \succ_s is much more discriminating than P -dominance \succ , which should improve pruning possibilities and fasten the search, this example shows that the determination of s -optimal solutions presents a new difficulty: we cannot construct a s -optimal path from s -optimal sub-paths. In other words, the Bellman principle [1] does not hold. Moreover, the compromise search problem is NP-hard. This can easily be derived from a result of Yu and Yang in [16] which proves that the Robust Deviation Shortest Path problem (RDSP for short) is NP-hard. In RDSP the goal is indeed to determine solution-paths minimizing function: $s(p, r) = \max_{i \in Q} \{ |p_i - r_i| \}$. Optimizing our function $s(p, r, \omega)$ with equal weights $\omega_i, i = 1, \dots, q$ clearly reduces to RDSP.

3 ALGORITHMS

Knowing that solutions of the compromise search problem lie in the Pareto set, we first recall how Pareto-optimal paths are generated by MOA* algorithm.

3.1 Using MOA* for compromise search

Algorithm MOA* is a heuristic search algorithm generating all efficient paths of G . It relies on the same foundations than A^* , adapted to the field of vector-valued evaluation. In particular, the Bellman principle holds: any s -to- k subpath of a P -non-dominated solution-path, is P -non-dominated in $\mathcal{P}(s, k)$. Hence, the algorithm constructs incrementally the P -non-dominated solution-paths from Pareto optimal subpaths. Let us briefly recall some essential features of MOA* (for more details, see [13]).

In a graph valued by cost-vectors, there possibly exists several P -non-dominated paths to reach a given node. Hence, at each detected node n , one stores a set $G(n)$ of cost-vectors $g(n)$ corresponding to the valuations of P -non-dominated paths, among the set of already detected path arriving in n . Thus, $G(n)$ is an approximation of the set $G^*(n)$ of all P -non-dominated cost vectors of paths in $\mathcal{P}(s, n)$. Moreover, the algorithm uses a vector-valued heuristic. Since a path arriving at node n can be continued with different incomparable subpaths to reach Γ , a set $H(n)$ of heuristic cost-vectors $h(n)$ is assigned to each node n . Thus, $H(n)$ is an approximation

of $H^*(n)$, the set of all P-non-dominated cost vectors of paths in $\mathcal{P}(n, \Gamma)$. As in A^* , already detected nodes of the graph are partitioned, at any step of the search, in two sets called OPEN (non-expanded nodes) and CLOSED (already expanded nodes able to be expanded again). During the search, for any open node n , the set $F(n) = \text{ND}(\{g(n) + h(n) : g(n) \in G(n), h(n) \in H(n)\})$ is computed from all possible combinations $f(n) = g(n) + h(n)$. Such vectors are called the *labels* of node n . Thus $F(n)$ is a set of labels (vectors) approximating the set $F^*(n)$ of all P-non-dominated cost-vectors containing node n .

The principle of the search is, at the current step, to expand a node n in OPEN such that $F(n)$ contains at least one P-non-dominated label among the labels of open nodes. The process is kept running until all open labels are either P-dominated by another label at the same node or by a vector solution-path already detected. It ensures to generate all efficient paths in $\mathcal{P}(s, n)$ provided heuristic H is admissible, i.e. $\forall n \in N, \forall h^* \in H^*(n), \exists h \in H(n) \text{ s.t. } h \succ h^*$.

Knowing that s -optimal solutions are also P-non-dominated solutions, the best compromise solution-path with respect to the Tchebycheff criterion s should be sought within the Pareto set. Hence, a straightforward way of determining best compromise solutions is 1) use MOA* to generate the Pareto set, and 2) determine the s -optimal solutions in the output of MOA*. Such a two-stages procedure should not be very efficient in practice due to the size of the Pareto set. For this reason, we introduce now two algorithms aiming at directly focusing on good compromise solutions.

3.2 BCA*

BCA* is a variant of MOA* aiming at finding a best compromise path, i.e. a solution-path P minimizing $s(v(P), r, \omega)$ over $\mathcal{P}(s, \Gamma)$ for a predefined r and ω . The basic principle is to work on P-non-dominated sub-paths while managing a current upper-bound λ to store the value $s(v(P), r, \omega)$ of the best compromise path P among already detected paths.

A preliminary stage consists of performing q monodimensional optimizations (one per criterion) with A^* so as to get one optimal path P^i for each criterion $i \in Q$. If $p^i = v(P^i)$ for all i , the ideal point α is given by: $\alpha_i = p^i$ for all $i \in Q$. The nadir point β is difficult to find in practice because the Pareto set is not known [3]. For this reason we use a classical approximation obtained by setting: $\beta_i = \max_{j \in Q} p_j^i$ for all $i \in Q$, whichs allows to compute weights $\omega_i = \delta_i / (\beta_i - \alpha_i)$, $i \in Q$. Then the choice of a given reference point r specifying the target in the space of criteria completely defines function s (by default choose r close to α , see property P2). Then the value of λ is initialized to $\lambda_0 = \min\{s(p^i), i \in Q\}$. Hence the main features of BCA* are:

Output: the best compromise solution-paths according to s .

Principle: as in [9], we expand labels (attached to subpaths) rather than nodes. We distinguish g -labels (without heuristic h) from f -labels (those used in MOA*). The search is totally ordered by function s . Thus, for any open node n and any f -label in $F(n)$ we compute the score $s(f(n))$ and we expand one of the f-labels having minimal score (the more promising compromise subpath).

Pruning: We use the following pruning rules: 1) at node n , we prune any subpath with a g -label P-dominated by the g -label of another subpath at the same node n . 2) we prune any subpath P at node n if its f -label $f(n)$ is such that $s(f(n)) > \lambda$. This second rule allows an early elimination of subpaths that cannot lead to the best compromise solutions. These rules are justified by the following proposition:

Proposition 1 $\forall x, y \in \mathbb{N}^q, x \succ y \Rightarrow x \succ_s y$. Moreover $\forall x, y, z \in \mathbb{N}^q, (x \succ_s y \text{ and } y \succ z) \Rightarrow x \succ_s z$.

Indeed, during the search, if a subpath P is P-dominated by another subpath P' at the same node n according to g -labels, any extension of P will be P-dominated by the same extension of P' , and therefore s -dominated. Similarly, if a subpath P is such that $f(v(P)) > \lambda = v(P^*)$, P^* being the best already detected solution-path, any extension of P will also be s -dominated by P^* .

Heuristics: we use any admissible heuristic that can be used in MOA*. In particular, we do not need to require that for all nodes n , $H(n)$ contains at least one perfectly informed label, as it is done to justify U^* [14].

Stopping condition: Each time a node of Γ is reached with a new path P such that $\lambda > s(v(p), r, \omega)$, the bound λ is updated. The algorithm is kept running until all open labels of type $f(n)$ satisfy $s(f(n)) > \lambda$. Termination obviously derives from the termination of MOA*. In worst case, BCA* performs the same enumeration as MOA* and finds the best compromise at the last step. Of course, in average, BCA* stops much earlier, as we shall see in the next section.

3.3 kA*

The second algorithm, called kA*, relies on the enumeration of k best paths in a scalarized version G' of the multiobjective graph G . Scalarization is obtained by considering the valuation $w : A \rightarrow \mathbb{R}$ derived from v by setting $w(a) = s_1(v(a), \omega)$ for all $a \in A$ (linear aggregation of criteria). Hence, any path P initially valued by a vector $p \in \mathbb{N}^q$ is now valued by the scalar cost $w(P) = s_1(p, \omega)$. Actually, using function s_1 is interesting even if its optimization does not provide directly compromise solutions. The point is that s -optimal solutions often achieve a good performance according to s_1 . Hence, we propose to enumerate solution-paths in G by increasing order of s_1 , which amounts to enumerating solution-paths in G' by increasing order of w . During enumeration, the s -optimal path can be identified thanks to the following proposition:

Proposition 2 $\forall x, y \in \mathbb{N}^q, s_1(y, \omega) > s_1(\bar{x}_\omega, \omega) \implies s(y, r, \omega) > s(x, r, \omega) \text{ with } \forall i \in Q, \bar{x}_{\omega_i} = r_i + \frac{1}{\omega_i} s(x, r, \omega)$.

Indeed, denoting P^1, \dots, P^j the first solution-paths in the enumeration, we have $s_1(p^1, \omega) \leq \dots \leq s_1(p^j, \omega)$. Suppose that P^j is the first of these paths satisfying $s_1(p^j, \omega) > s_1(p_\omega^*, \omega)$ where $p^* = \arg \min_{i=1, \dots, j} s(p^i, r, \omega)$ is the cost of the current best path P^* , then proposition 2 implies that all forthcoming paths $P^k, k > j$ in the enumeration will satisfy $s(p^k, r, \omega) > s(p^*, r, \omega)$. Therefore P^* is the best compromise solution-path with respect to s and we can stop the enumeration. Hence the main features of kA* are:

Output: the best compromise solutions according to s .

Principle: The ordered enumeration of best paths according to s_1 can easily be performed using a modified version of A^* that works on labels attached to paths instead of labels attached to nodes. More precisely, to any detected path $P_{s,n}^i$ from s to n , we assign a label of type $f^i(n) = g^i(n) + h(n)$ where $g^i(n) = w(P_{s,n}^i)$ and $h(n)$ is the value of an admissible heuristic function at node n ($\forall n \in N, h(n) \leq h^*(n)$ where $h^*(n)$ is the cost of the optimal path in $\mathcal{P}(n, \Gamma)$). Hence, we have as many labels at a given node n as detected sub-paths from s to n . Here $G(n)$ (resp. $F(n)$) denotes the set of g -labels (resp. f -labels) of detected sub-paths from s to n . So we have $F(n) = \{f^i(n) = g^i(n) + h(n), g^i(n) \in G(n)\}$. At node n , $f^{(i)}(n)$ denotes the i^{th} smallest value in $F(n)$. At the current step, we expand the label $f^{(1)}(n) = \min_{t \in \text{OPEN}} f^{(1)}(t)$. Since $f^{(1)}(n)$ is also the best current f -label in $F(n)$, there is no need to expand

another label of $F(n)$ before the best solution-path corresponding to $f^{(1)}(n)$ is developed. Therefore n is put in CLOSED. If n is a goal node then the k^{th} path P^k is found. In this case, for all nodes t along P^k , we delete the label $f^{(1)}(t)$ in $F(t)$. In order to expand the new $f^{(1)}$ -label of these nodes, we set all these nodes t in OPEN. If $s(p^k, r, \omega) < s(p^*, r, \omega)$ (where p^* is the best detected cost vector) and we set $P^* \leftarrow P^k$.

Heuristics: any type of admissible scalar-valued heuristic, e.g., any h derived by linear scalarization of an admissible heuristic used in MOA* since s_1 is strictly monotonic with respect to P-dominance.

Stopping condition: thanks to proposition 2, path enumeration is kept running until a solution-path P^k such that $s_1(p^k, \omega) > s_1(\bar{p}_\omega^*, \omega)$ (where P^* is the best current path) is detected.

4 EXPERIMENTATIONS

Numerical experiments have been performed to compare the three solutions discussed in the paper for compromise search: MOA*(completed with the determination of s -optimal solutions in the Pareto-set) BCA* and kA*. The computational experiments were carried out with a Pentium IV with 2.5GHz processor. Algorithms were implemented in C++. We first present result on randomly generated instances and then on pathological instances.

Randomly generated graphs. We relate here results obtained by executing algorithms on randomly generated graphs. The number of nodes in these graphs varies from 200 to 1,000 and the number of arcs from 7,500 (for 200 nodes) to 200,000 (for 1,000 nodes). Cost vectors are randomly determined for each arc. In practice, good admissible heuristics depend on the application domain. Independently of the context, we can generate admissible heuristic for tests, by setting $H(n) = \mu(n)H^*(n)$ for vector valued optimization, where $\mu(n)$ is a randomly generated number within the unit interval at node n . For kA*, the heuristic is obtained from the previous one by linear scalarization. Table 1 presents the average execution times (in second) obtained by the three algorithms for different class differing by the size of instances (number of nodes, number of criteria). In each class, average times are computed over 50 different instances. Symbol “-” means that execution time exceeds one hour.

Table 1. Execution time (s) on Random Graphs

#obj.	MOA*			BCA*			kA*		
	5	10	20	5	10	20	5	10	20
200	0.1	8.2	583	0	0	0.1	0	0	2.2
500	3.3	506	-	0.1	0.3	1.5	0.1	0.3	4.5
800	8.3	-	-	0.2	1.1	7.9	0.3	1.1	5.7
1,000	15.2	-	-	0.2	1.6	13	0.7	1.2	6.3

These results show that BCA* and kA* determine the s -optimal solution paths significantly faster than MOA*. Indeed, for a graph containing more than 500 nodes and 10 criteria, MOA* requires more than one hour to generate the s -optimal solutions whereas BCA* and kA* need some seconds. We see that the sophistication proposed in BCA* really speed-up the search of the best compromise solutions. Moreover, we can notice that BCA* is slightly faster than kA* when the size of the graph is small, but kA* outperforms BCA* as the size of the instance increases.

Hansen graphs. Now, algorithms are compared on Hansen graphs (Figure 1). All feasible paths in this family of graphs are P-non-dominated. Then MOA* generates all solution-paths. Since for any path P in Hansen's family, we have $s_1(\bar{p}_\omega^*, \omega) > s_1(p, \omega)$ (where P^* is the best compromise path), kA* generates all feasible paths too. Table 2 summarizes execution times obtained on these graphs.

Table 2. Execution time (s) on Hansen Graphs

#nodes	MOA*	BCA*	kA*
21	0.01	0	0
25	3.1	1.1	0
29	70.2	25.5	0.15
33	1147	427.9	0.61

These results show the advantage of resorting to a scalarized version of the graph. Indeed kA* reveals very efficient when the number of P-non-dominated paths is important, because scalarization avoids numerous Pareto-dominance tests.

5 CONCLUSION

We have introduced two admissible algorithms to determine the best compromise paths in multivalued state space graphs. Although they produce the same solutions, BCA* is based on a focused exploration of the Pareto-set in a multiobjective space while kA* solves the same problem by working on a scalarized version of the graph. As expected, tests confirm that the two approaches are much more efficient than explicit enumeration of the Pareto set to solve the compromise search problem. An advantage of the approach used in kA* is that it applies to any other multiobjective problem, provided the scalar version can be solved efficiently. Both algorithms perform a fast generation of the best current compromise solution and could be incorporate into an interactive exploration procedure of the Pareto set.

REFERENCES

- [1] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
- [2] P. Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar, ‘Utility of pathmax in partial order heuristic search’, *J. of algorithms*, **55**, 317–322, (1995).
- [3] M. Ehrgott and D. Tenfelde-Podehl, ‘Computation of ideal and nadir values and implications for their use in mcdm methods’, *European Journal of Operational Research*, **151**(1), 119–139, (2003).
- [4] P. Hansen, ‘Bicriterion path problems’, in *Multiple Criteria Decision Making: Theory and Applications, LNEMS 177*, eds., G. Fandel and T. Gal, 109–127, Springer-Verlag, Berlin, (1980).
- [5] P. E. Hart, N. J. Nilsson, and B. Raphael, ‘A formal basis for the heuristic determination of minimum cost paths’, *IEEE Trans. Syst. and Cyb. SSC-4* (2), 100–107, (1968).
- [6] U. Junker, ‘Preference-based search and multi-criteria optimization.’, in *AAAI/IAAI*, pp. 34–40, (2002).
- [7] P. Kouvelis and G. Yu, *Robust discrete optimization and its applications*, Kluwer Academic Publisher, 1997.
- [8] B. Logan and N. Alechina, ‘A* with bounded costs’, in *Proceedings of AAAI-98*. AAAI Press/MIT Press, (1998).
- [9] L. Mandom and J.L. Prez de la Cruz, ‘A new approach to multiobjective a* search’, in *Proceedings of IJCAI-05*, pp. 218–223. Professional Book Center, (2005).
- [10] P. Perny and O. Spanjaard, ‘Preference-based search in state space graphs’, in *Proceedings of AAAI-02*, pp. 751–756, (2002).
- [11] P. Perny and O. Spanjaard, ‘An axiomatic approach to robustness in search problems with multiple scenarios’, in *Proceedings of UAI*, pp. 469–476, Acapulco, Mexico, (8 2003).
- [12] R.E. Steuer and E.-U. Choo, ‘An interactive weighted Tchebycheff procedure for multiple objective programming’, *Mathematical Programming*, **26**, 326–344, (1983).
- [13] B.S. Stewart and C.C. White III, ‘Multiobjective A*’, *Journal of the Association for Computing Machinery*, **38**(4), 775–814, (1991).
- [14] C.C. White, B.S. Stewart, and R.L. Carraway, ‘Multiobjective, preference-based search in acyclic OR-graphs’, *European Journal of Operational Research*, **56**, 357–363, (1992).
- [15] A.P. Wierzbicki, ‘On the completeness and constructiveness of parametric characterizations to vector optimization problems’, *OR Spektrum*, **8**, 73–87, (1986).
- [16] G. Yu and J. Yang, ‘On the robust shortest path problem’, *Comput. Oper. Res.*, **25**(6), 457–468, (1998).

MINION: A Fast, Scalable, Constraint Solver¹

Ian P. Gent² and Chris Jefferson³ and Ian Miguel²

Abstract. We present MINION, a new constraint solver. Empirical results on standard benchmarks show orders of magnitude performance gains over state-of-the-art constraint toolkits. These gains increase with problem size – MINION delivers *scalable* constraint solving. MINION is a general-purpose constraint solver, with an expressive input language based on the common constraint modelling device of matrix models. Focussing on matrix models supports a highly-optimised implementation, exploiting the properties of modern processors. This contrasts with current constraint toolkits, which, in order to provide ever more modelling and solving options, have become progressively more complex at the cost of both performance and usability. MINION is a black box from the user point of view, deliberately providing few options. This, combined with its raw speed, makes MINION a substantial step towards Puget’s ‘Model and Run’ constraint solving paradigm.

1 Introduction

Constraint programming supports the solution of combinatorial problems in two stages. First, by characterising or *modelling* them by a set of constraints on decision variables that their solutions must satisfy. Second, by searching for *solutions* with a constraint solver: assignments to the decision variables that satisfy all constraints. The success of constraint programming is based upon its flexibility and expressivity; constraints provide a rich language allowing most problems to be expressed straightforwardly once the model is selected.

Modern constraint solving techniques are typically provided to users in the form of constraint toolkits. These are libraries for programming languages such as C++ (e.g. ILOG Solver, GeCode), Java (e.g. Choco, Koalog) or Prolog (e.g. Eclipse, Sictus). This approach supports a high degree of customisation. For example, it is usually possible for users to add propagation rules for an as-yet undefined constraint to the library. When such additions prove to be generally useful, they are often included in future versions of the popular constraint toolkits. Hence, as Puget pointed out [20], constraint toolkits have become increasingly complex in an effort to provide ever more functionality. The result is that constraint programming for large, realistic problems requires a great deal of expertise and fine tuning. Furthermore, to provide this flexibility,

toolkits have a necessarily complex internal architecture. This results in large overheads in many parts of the solver, with a detrimental effect on efficiency and scalability.

Puget suggests that today’s constraint toolkits are far too complex to achieve widespread acceptance and use [20]. He advocates a ‘model and run’ paradigm for constraints similar to that achieved by propositional satisfiability (SAT) and mathematical programming (MP). Our hypothesis is that we can achieve this by taking two important steps. The first step is to supply constraint *solvers* (instead of toolkits) that accept *models* rather than *programs*. These solvers should be black boxes that reduce the bewildering array of options provided by constraint toolkits to a select few. Furthermore, the reliance of constraint programming on heavy fine tuning to achieve an acceptable level of performance is in conflict with ‘model and run’. The second step is therefore to create *fast* constraint solvers that promote implementation efficiency. Efficiency in turn will reduce the sensitivity of performance to minute parameter changes. Fast solvers will also be more *scalable* to large constraint problems. This methodology follows the closely-related fields of propositional SAT and MP where solvers scale to very large problems, and where ‘model and run’ is absolutely standard.

To test our hypothesis, we have built a fast and scalable constraint solver, named MINION. Throughout, MINION has been optimised for solving *large* and *hard* problems. Experimental results show that MINION typically runs between 10 and 100 times faster than state-of-the-art constraint toolkits such as ILOG Solver and GeCode on large, hard problems. On smaller problems or instances where solutions are found with little search, gains are less impressive. MINION takes an expressive input language for matrix models over integer domains, allowing a variety of fundamental constraints such as all-different, sum, reification, and others. A number of our design decisions are modelled on those of zChaff, which revolutionised modern SAT solving [23]. In particular, we paid special attention to using very small data structures, making memory usage very small and greatly increasing speed on modern computer architectures.

2 Background

A *constraint satisfaction problem* (CSP [3]) is a set of decision variables, each with an associated domain of potential values, and a set of constraints. An *assignment* maps a variable to a value from its domain. Each constraint specifies allowed combinations of assignments of values to a subset of the variables. A *solution* is an assignment to all the variables that satisfies all the constraints. A *constrained optimisation problem* is a

¹ We thank Angela Miguel, Karen Petrie, Judith Underwood and ECAI’s referees. Supported by EPSRC Grant GR/S30580 and a Royal Academy of Engineering/EPSRC Research Fellowship.

² School of Computer Science, University of St Andrews, St Andrews, Fife, KY16 9SS, UK. {ipg,ianm}@dcs.st-and.ac.uk

³ Oxford University Computing Laboratory, University of Oxford, Oxford, UK. chris.jefferson@comlab.ox.ac.uk. Research undertaken while at St Andrews University.

CSP with an objective function, which must be optimised.

We focus on systematic search for solutions in a space of assignments to subsets of the variables. We use constraint *propagation* algorithms that make inferences, recorded as domain reductions, based on the domains of the variables constrained and the assignments that satisfy the constraint.

MINION uses a language based upon *matrix models*, i.e. CSP formulations employing one or more matrices of decision variables, with constraints typically imposed on the rows, columns and planes of the matrices. To illustrate, consider the *Balanced Incomplete Block Design* (BIBD, CSPLib problem 28), which is defined as follows: Given a 5-tuple of positive integers, $\langle v, b, r, k, \lambda \rangle$, assign each of v objects to b blocks such that each block contains k distinct objects, each object occurs in exactly r different blocks and every two distinct objects occur together in exactly λ blocks. The matrix model for BIBD has b columns and v rows of 0/1 decision variables. A ‘1’ entry in row i , column j represents the decision to assign the i th object to the j th block. Each row is constrained to sum to r , each column is constrained to sum to k and the scalar product of each pair of rows is constrained to equal λ . Matrix models have been identified as a very common pattern in constraint modelling [6] and support, for example, the straightforward modelling of problems that involve finding a function or relation — indeed, one can view the BIBD as finding a relation between objects and blocks.

3 Modelling in MINION

MINION’s input language is simple yet expressive enough to model the majority of problems without resorting to complex and expensive modelling devices. Critically, we include a range of the most important non-binary constraints as primitives. We do not go so far as restricting the input language to (for example) binary constraints in extensional form, which is sufficient to express any CSP. Reformulation into binary constraints can lead to weaker propagation and far more search.

MINION has four variable types. These are: 0/1 variables, which are used very commonly for logical expressions, and for the characteristic functions of sets; Bounds variables, where only the upper and lower bounds of the domain are maintained; Sparse Bounds variables, where the domain is composed of discrete values, e.g. $\{1, 5, 36, 92\}$, but only the upper and lower bounds of the domain are updated during search; and Discrete variables, where the domain ranges from the lower to upper bounds specified, but the deletion of any domain element in this range is permitted. (A fifth type, of Discrete Sparse variables, is not yet implemented.) Sub-dividing the variable types in this manner affords the greatest opportunity for optimisation, as we will see. MINION’s input language supports the definition of one, two, and three-dimensional matrices of decision variables (higher dimensions can be created by using multiple matrices of smaller dimension). Furthermore, it provides direct access to matrix rows and columns since most matrix models impose constraints on them.

The input language for MINION is detailed in Figure 1. The set of primitive constraints provided by MINION is small but expressive. It includes necessities such as *equality*, *disequality*, *inequality*, *sum*, *product*, and *table* (extensional form) but also global constraints that have proven particularly useful as reported in the literature over several years, such as

```

<MinionInput> ::= 
<noOf01Vars>
<noOfBoundsVars> {<lb> <ub> <number>}
<noOfSparseBoundsVars> {'{' <elem>{, <elem>}'}' <number>}
<noOfDiscreteVars> {<lb> <ub> <number>}
<variableOrder>
<valueOrder>
<noOf1dMatrices> {<literalVar1dMatrix>}
<noOf2dMatrices> {<literalVar2dMatrix>}
<noOf3dMatrices> {<literalVar3dMatrix>}
objective <objectiveExpression>
{<constraint>}

<objectiveExpression> ::= 
'none' | 'minimising' <var> | 'maximising' <var>

<constraint> ::= 
<reifiableConstraint> |
reify(<reifiableConstraint>, <var>) |
table(<varVectorExpression>, <tuples>)

<reifiableConstraint> ::= 
allDiff(<varVectorExpression>) |
<eqOrDiseqConstraint> (<var>, <varOrConst>) |
element(<varVectorExpression>, <var>, <varOrConst>) |
ineq(<var>, <var>, <const>) |
<lexConstraint> (<varVectorExpression>, <varVectorExpression>) |
<MinOrMaxConstraint> (<varVectorExpression>, <varOrConst>) |
occurrence(<varVectorExpression>, <const>, <var>) |
product(<varVectorExpression>, <varOrConst>) |
product(<varVectorExpression>, <literalConstVector>,
<varOrConst>) |
sum(<varVectorExpression>, <varOrConst>)

<varVectorExpression> ::= 
<literalVarVector> | <1dMatrixId> | <2dMatrixId> |
<3dMatrixId> | <rowOrCol>(<2dMatrixId>, <index>) |
<colOrRowXOrRowY>(<3dMatrixId>, <index>, <index>)

```

Figure 1. Syntax of MINION input. Although some non-terminals are unexpanded, their meanings should be clear.

all-different [21] and *occurrence*. We include the crucial *element* constraint, which allows one to specify an element of a vector using the assignment to a decision variable. This is often useful when *channelling* between matrix models [2]. Logical expressions are supported by using 0/1 variables with the *min* (conjunction), *max*, *sum* (both disjunction) and inequality (implication) constraints. *Reification* is to assign the satisfaction (1) or unsatisfaction (0) of a given constraint to a 0/1 decision variable. Since this decision variable can participate as normal in other constraints this adds significant expressive power to the language. We include the *lexicographic ordering* constraint, which has been shown to be a cheap and effective way to exclude much of the symmetry in a matrix model [8]. The current version of MINION uses ‘watched literals’ to implement the element, table, and 0/1 sum constraints. We believe this to be an important innovation, but we describe it in detail elsewhere [12]. MINION allows only static variable and value ordering heuristics. Dynamic heuristics could be implemented, but are currently omitted to avoid unnecessary overheads when *not* being used. The only design decisions taken for reasons of time were the omission of the global cardinality constraint, and that MINION’s all-different performs the same propagation that a clique of not-equals constraints would (though far faster). The implementation of some propagators in MINION is simplified by the use of views [22], which allow complex propagation algorithms to be built from simpler ones by applying simple transformations to the variables of the constraint.

MINION deliberately does not allow arbitrary operator nesting. Most toolkits support this feature by adding hidden decision variables to ‘flatten’ the input. For example, $x_1 * x_2 + x_3 = x_4$ might become $h_1 = x_1 * x_2$, $h_2 = h_1 + x_3$ and $h_2 = x_4$. The flattening scheme is often inaccessible to the user, and may not be optimal. Expecting pre-flattened input has two important consequences. First, it simplifies the input language. Second,

the user is not forced to use a flattening scheme chosen by us, but can choose one most appropriate to the model. Removing this layer of hidden variables is especially important with the advent of automated constraint modelling systems such as CONJURE [9] that, in order to choose among candidate models, need maximum control over the final model.

4 Architecture and Design of MINION

Our principal challenge was to develop a constraint solver that is optimised to the same degree as those found in SAT (e.g. zChaff [23]) and mathematical programming (e.g. CPLEX [14]). A main design principle is compactness. This lets us take advantage of modern cache-based hardware architectures, which has paid substantial dividends in the SAT community [25]. Further, modern processors attempt to execute many instructions in parallel, and so branching, indirection and the use of offsets should be avoided where possible, as they can stall this parallel execution. Many design decisions were made after extensive code profiling. For example, profiling shows that there are many more calls to access variable domains than there are to change them. We thus optimise data structures for variables to make access fast at the expense of slower domain updates, even where this conflicts with keeping data structures small.

This section includes ideas we believe to be novel. It also includes description of numerous implementation techniques, not contributions to knowledge in themselves, which are important in obtaining our fast running times. MINION is the experimental apparatus with which we are testing our overall hypothesis, that fast constraint solvers can be built. It is thus essential that MINION is described in detail, so that our work can be scientifically judged and also reproduced and built on. In this spirit, we have made MINION open source, and it is available at <http://minion.sourceforge.net>.

Variable representation: Variables' values have to be restored on backtracking, so the smaller the space they occupy, the less work has to be done. Also, a smaller block of memory is more likely to fit into L2 cache on a modern CPU, greatly increasing access speed. In each of the four types of variable, storage is split into two: a *backtrackable* and a *non-backtrackable* part. This reduces backtrackable memory usage, and thus the amount of work on backtracking. To the best of our knowledge, these representations are novel.⁴

We use two bits to store the state of a 0/1 variable's domain. However, it is unnecessary to restore both bits on backtracking. Bit 1 indicates whether the variable is assigned. Bit 2 gives the value *if it is assigned*. Before search, bit 1 is 0 and bit 2 is arbitrary. When the variable is assigned, bit 1 is set to 1, while bit 2 is given the correct value. When backtracking past this point, we reset bit 1 to 0, but do not restore bit 2. Bit 2's value is irrelevant until the variable is reassigned, when it will be set to 0 or 1 irrespective of its current value. Hence, we maintain k 0/1 variables by saving and restoring a block of just k bits at each choice point. In non-backtrackable memory, we store the second block of k bits.

Next, consider bounds variables. Suppose a variable is declared with non-sparse bounds $[lb, ub]$. We store the initial

lower bound lb in non-backtrackable memory. In backtrackable memory we store the current bounds relative to lb , initialised to $[0, ub - lb]$. The amount of backtrackable memory required is therefore twice the smallest C++ type which will store the initial domain size $ub - lb$. For sparse bounds variables, the storage in backtrackable memory is identical. In non-backtrackable memory, we store an array containing the initial domain. When, say, a request is made to increase the lower bound, we undertake a binary search for the smallest allowed value which is greater than or equal to the new bound, and the backtrackable lower bound is set to this value.

For discrete variables, we store in backtrackable memory (as is conventional) one bit per value, indicating whether or not the value is still in the domain. We also store the current lower and upper bounds of the domain similarly to bounds variables. This redundant storage conflicts with our key principle of minimising backtrackable memory, but is justified because many constraints check only the bounds of domains. For example, by profiling we found that Choco can spend 33% of its time finding the bounds of domains. Our technique also lets us optimise updating the bounds of discrete domains: we do *not* take the linear time required to zero the relevant parts of the bit array. Instead, we adapt the check for domain membership. The bit array is only checked if the domain value is between the upper and lower bounds, so bits outside that range are irrelevant. This slightly improves the speed of domain checks when the value proves to be outside the current range, but slows down those where it is in range. Bit arrays are stored in blocks of 8, 16, 32, or 64 bits as these can be efficiently stored and accessed while reducing wasted space.

Finally, MINION implements a special constant variable type, which is assigned a single value and uses no backtrackable memory. For most constraints in MINION, the compiler is able to optimise code using this variable type as efficiently as an implementation of the constraint designed to take advantage of the fact the variable is known to be constant.

Memory Management: Variables of each type are stored together in backtrackable memory. For example, all Boolean variables are stored in a single bit array. Thus, 1,088 Boolean variables would need 136 bytes of backtrackable memory. Boolean variables are accessed during search with three values computed at initialisation: two pointers to the words containing the bit in backtrackable and non-backtrackable memory, and a mask that contains a 1 at the relevant bit position in those words and is otherwise 0. This lets us access the relevant bits in a small number of machine instructions. Other variable types are compressed similarly. For example, on CPUs with 64-bit words, each word stores the domains of four discrete domain variables with 16 values, with lower and upper bounds stored elsewhere in backtrackable memory. After backtrackable variables, we allocate storage for backtrackable data needed by constraint propagators. We have blocks of space for binary flags, integers up to 255, up to 65,535, and unstructured space for backtrackable data not in one of these forms. By design, we have minimised the fragmentation in this memory map, although this leads to an initialisation overhead. We allocate space in two passes: first we place variables arbitrarily, but when all variables and constraints have been constructed, we then rearrange memory as described above.

State restoration on backtracking is simple. At a choice point, we copy the entire block of backtrackable memory.

⁴ It remains possible that some have escaped our notice in publications or simply used in solvers not described in publications.

When we return to that point, we copy the entire archived copy over the active backtrackable memory. Block copying is sped up considerably because all backtrackable memory is allocated together and because we have minimised the size of backtrackable memory so much. In our preliminary investigations memory copy did not rise above 7% of runtime. In BIBD experiments we report below, as much as 750 MB of backtrackable memory was copied per second. This shows both the capabilities of modern processors, and also the importance of keeping memory sizes small. We could save 50% of memory copies by redirecting all pointers to the stored copy on backtracking, saving the memory copy on backtracking. However, this would not let us use fixed pointers for each variable. We trade extra copying against reduced pointer arithmetic. Many techniques can reduce the amount of memory changed on backtracking, e.g. dancing links [17], but we have not found one that repays even low time overheads.

Low-level Optimisations: Most MINION constraints accept any variable type. Traditionally such functions are implemented via virtual function calls, which select the correct function at run-time based on the variable type. Instead, we generate a copy of the constraint at compile-time for each variable type using C++ templates. The compiler can then optimise each constraint for each variable type, inlining small domain access methods and avoiding run-time branches depending on variable type. On the $\langle 7, 315, 135, 3, 45 \rangle$ BIBD for example, compiling each constraint for the Boolean type provides a fourfold speed-up. This method limits the number of variable types per constraint. For example, the lexicographic ordering constraint allows only one variable type per input vector to avoid exponential explosion in the copies of the constraint that must be compiled. This limitation is not serious. In most CSPs each constraint contains only one or two variable types — but we do provide a fall-back implementation of each constraint for arbitrary combination of types via virtual function calls. While the compiler optimises each constraint for different variable types, it may not be able to perform all possible optimisations. For example when the lower bound of a Boolean variable changes, the variable must now be assigned 1, but the compiler does not identify this. We have identified and implemented optimisations for some such special cases.

MINION constraints contain pointers to the locations of their variables in memory. This uses more memory than storing this information once globally for each variable, and initialisation requires storing the locations of all references to variables as the problem is constructed. This accounts for the higher initialisation cost of MINION than, say, ILOG Solver, but allows extremely fast variable access during search.

5 Empirical Analysis

We show that MINION outperforms state-of-the-art constraint solvers (ILOG Solver, Eclipse and GeCode), and scales to much larger instances. We consider a portfolio of instances from diverse, challenging problem classes: a satisfaction problem (BIBD), two optimisation problems (Steel Mill Slab Design [11] and Golomb Rulers [4]) and a fixed-length planning problem (English Peg Solitaire [16]). Our goal was to compare raw search speed on identical models.

We compared implementations in each system of the same model: that of the BIBD described earlier, for Steel Mill and Peg Solitaire the basic models from the cited papers, and for

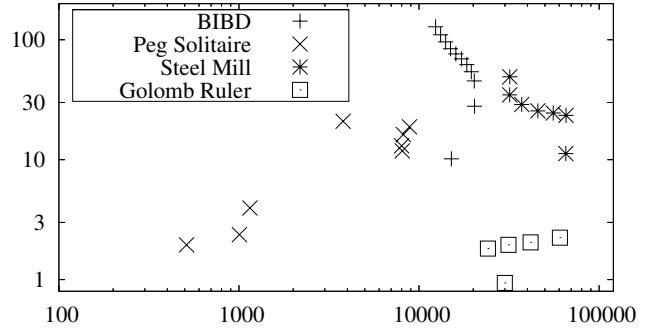


Figure 2. MINION Backtracks/Sec. (x axis) vs Speedup against ILOG Solver (y axis) on four problem classes

Golomb a basic model using simple all different propagation. We used straightforward implementations in each solver of the chosen model, with the same static variable and value orderings in each case.⁵ Our generators for MINION instances are part of the source distribution. We used revision r170 of MINION, ILOG Solver 5.3, and GeCode 1.1.0. Experiments ran under Windows XP SP2, on a Pentium 4 3GHz with 2GB RAM, compilation being done using g++ 3.4.4 under cygwin (and Visual C++ for Solver.) We omit detailed results for Eclipse, as it was slower than ILOG Solver in each case.

Figure 2 and Table 1 summarise our results. These show that MINION is faster and much more scalable than other state-of-the-art solvers. Our results show particularly dramatic improvements on large constraint problems requiring a lot of search. MINION ran faster than ILOG Solver on every instance except the smallest Golomb Ruler instance ($n = 7$), which both solved in less than 0.02sec. Our least impressive gains are on the Golomb Ruler, which are small problems, where we were 1.8 times as fast on the largest instance we tested ($n = 11$). Speedups were also relatively low on the simplest Solitaire instances. These result from MINION’s greater initialisation time, a part of its design which has not been optimised highly but which is important for instances requiring more search. On Solitaire instances where MINION needs at least 2sec. to solve, it is at least 10 times faster than Solver. For very large instances where much search is required, MINION’s speedup reaches 128 for the largest BIBD instance and 49 for the largest Steel Mill instance. Where we compared against GeCode, results were similar except that GeCode was usually faster than ILOG Solver. Again we were faster on each Golomb instance but $n = 7$, but gains were even more marginal here, e.g. a negligible 2.9% on $n = 11$. However, Table 1 shows increasing speedups with problem size on BIBD instances, up to 51 times on the largest we tested.

It is possible that greatly improved runtimes might be obtained in GeCode or Solver using the flexibility that each toolkit provides. Non-optimal behaviour on some instances goes hand-in-hand with using the ‘model and run’ methodology, so we should point out some of its advantages. First, improvements in *modelling* can be duplicated in MINION. Other improvements might require significant expertise in the relevant toolkit, such as implementing specialised constraints or heuristics. Second, a 10-fold speedup obviates the need for much optimisation. For example, finding five independent op-

⁵ We were unable to compare GeCode and MINION directly on the Steel Mill and Peg Solitaire problems because of the differing modelling facilities offered by each system at the time of writing.

Balanced Incomplete Block Design								Steel Mill			Peg Solitaire		
v b r k λ	Minion		ILOG Solver		GeCode		orders	Minion BT/s	Solver BT/s	Minion sec.	ILOG Solver sec.	Peg Solitaire BT/s	
	sec.	BT/s	sec.	BT/s	sec.	BT/s							
7 140 60 3 20	1.1	15 000	12	1 500	12	1 400	40	65 000	5 800	1.3	510	2.5	260
7 210 90 3 30	3.3	20 000	92	730	75	890	50	66 000	2 800	1.4	1 000	3.3	420
7 280 120 3 40	9.0	20 000	410	450	280	650	60	56 000	2 300	1.4	1 200	5.5	290
7 315 135 3 45	14	20 000	770	360	500	560	70	46 000	1 800	2.4	3 800	50	180
7 350 150 3 50	22	18 000	1 400	300	920	440	80	37 000	1 300	5.7	8 200	92	510
7 385 165 3 55	33	17 000	2 300	250	1 400	420	90	32 000	920	33	8 000	430	610
7 420 180 3 60	49	16 000	3 600	210	2 200	350	100	32 000	650	57	8 100	680	680
7 455 195 3 65	71	15 000	3 600	180	3 300	320				130	8 800	2 400	470
7 490 210 3 70	100	14 000	3 600	150	4 700	300							
7 525 225 3 75	140	13 000	3 600	120	6 200	290							
7 560 240 3 80	190	12 000	3 600	96	9 500	240							

Table 1. Times and Backtracks per Second on 3 Problem Classes. Results given to 2 significant figures. Times in italics indicate timeouts of ILOG Solver. For steel mill instances both solvers timed out after 600s on every instance.

timisations to reduce run time by a third is less effective than a 10-fold speedup since $\frac{2}{3}^5 > \frac{1}{8}$. Finally, by making MINION open source, we are helping other researchers either to implement key techniques in MINION, or to use our insights to improve their own solvers or toolkits.

6 MINION and Modelling Languages

Although MINION’s input language is expressive, by design it is free from syntactic sugar to aid human modellers. The intention is that it will be used with constraint modelling languages that prioritise expressing models naturally. One popular example is the *Optimization Programming Language* (OPL [24]). We believe that building an OPL to MINION translator will be straightforward. In cases such as set variables, which OPL has but MINION does not, encodings into more primitive variables are well known [15]. By a similar process, translators to MINION could also be written for other constraint modelling languages, such as Constraint Lingo [5] and EaCL [18]. The abstract constraint modelling languages ESRA [7] and F [13], which support relation and function variables respectively, have translators to OPL. This supports a two-stage translation to MINION, given an OPL to MINION translator. NP-SPEC [1] also supports abstract decision variables, such as sets, permutations, partitions and functions. Currently NP-SPEC specifications are translated into SAT. This translator could be modified to produce MINION models instead.

MINION should combine very well with the CONJURE automated modelling system. Given an abstract specification of a constraint problem in the ESSENCE language [10], a set of rules can formalise the generation of alternative models [9]. These rules are embedded in CONJURE, which *refines* an ESSENCE specification into an ESSENCE’ model. ESSENCE’ is a generic constraint language, straightforwardly translated into the input for ILOG Solver, Eclipse, or MINION. Although alternative models can be produced automatically, there is as yet no mechanism for model selection in CONJURE. Model selection must consider both model *and* solver. Given the transparent nature of MINION – without hidden modelling as discussed in Section 3 – model selection should be significantly simpler.

7 Conclusion and Future Work

MINION is a strong platform for future constraints research in modelling and solving. Once constraint modelling languages interface to it, researchers should be able to get a much better idea of how well a particular model performs, without fear of failing to exploit a given toolkit optimally. We have

described a methodology for building fast and scalable constraint solvers. While we chose a particular feature set and optimised accordingly, similar design principles should help to build very different solvers: for example, fast and scalable solvers using backjumping [19] and learning.

REFERENCES

- [1] M. Cadoli, G. Ianni, L. Palopoli, A. Schaerf, D. Vasile. NP-SPEC: An Executable Specification Language for Solving all Problems in NP. *Computer Languages* 26, 165-195, 2000.
- [2] B. M. W. Cheng, K. M. F. Choi, J. H-M. Lee, J. C. K. Wu. Increasing Constraint Propagation by Redundant Modeling: an Experience Report. *Constraints* 4(2), 167-192, 1999
- [3] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.
- [4] A.K. Dewdney. Computer Recreations. *Scientific American*, December, 16-26, 1985.
- [5] R. A. Finkel, V. W. Marek, M. Truszczynski. Constraint Lingo: towards high-level constraint programming. *Software - Practice and Experience*, 34 (15), 1481-1504, 2004
- [6] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh. Matrix modelling: Exploiting common patterns in constraint programming. *Int. Wshop on Reformulating CSPs*, 227-41, 2002.
- [7] P. Flener, J. Pearson, M. Agren. Introducing ESRA, a relational language for modelling combinatorial problems. *LOPSTR*, 214-232, 2004.
- [8] A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh. Global Constraints for Lexicographic Orderings. *CP*, 93-108, 2002.
- [9] A.M. Frisch, C. Jefferson, B. Martinez-Hernandez, I. Miguel. The Rules of Constraint Modelling. *IJCAI*, 109-116, 2005.
- [10] A.M. Frisch, M. Grum, C. Jefferson, B. Martinez-Hernandez, I. Miguel. The Essence of ESSENCE. *4th Int. Wshop. on Modelling and Reformulating CSPs*, 73-88, 2005.
- [11] A.M. Frisch, I. Miguel, T. Walsh. Symmetry and Implied Constraints in the Steel Mill Slab Design Problem. *Int. Wshop. on Modelling and Problem Formulation*, 8-15, 2001.
- [12] I.P. Gent, C. Jefferson, I. Miguel. Watched Literals for Constraint Propagation in Minion. CP-Pod Report 17-2006, 2006.
- [13] B. Hnich. *Function Variables for Constraint Programming*. PhD Thesis, Uppsala University, 2003.
- [14] ILOG. *ILOG CPLEX 9.0 Reference Manual*, ILOG, 2005.
- [15] C. Jefferson, A.M. Frisch. Representations of Sets and Multisets in Constraint Programming. *4th Int. Wshop. on Modelling and Reformulating CSPs*, 102-116, 2005.
- [16] C. Jefferson, A. Miguel, I. Miguel, S. A. Tarim. Modelling and Solving English Peg Solitaire. *Computers and Operations Research*, 33(10), 2935-2959, 2006.
- [17] D.E. Knuth. Dancing Links. *Millennial Perspectives in Computer Science*, Palgrave, 187-214, 2000.
- [18] P. Mills, E. Tsang, R. Williams, J. Ford, J. Borrett. *EaCL 1.5: An Easy Abstract Constraint Programming Language*. Technical Report, University of Essex, 1999.
- [19] P. Prosser. Hybrid Algorithms for the Constraint Satisfaction Problem. *Computational Intelligence* 9, 268-299, 1993.
- [20] J-F. Puget. Constraint Programming Next Challenge: Simplicity of Use. *CP*, 5-8, 2004.
- [21] J-C. Regin. A Filtering Algorithm for Constraints of Difference in CSPs. *AAAI*, 362-367, 1994.
- [22] C. Schulte, G. Tack. Views and Iterators for Generic Constraint Implementations. *CICLOPS*, 37-48, 2005.
- [23] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: engineering an efficient SAT solver. *DAC*, 530-535, 2001.
- [24] P. van Hentenryck. *The OPL Optimization Programming Language*. MIT Press, Cambridge, Massachusetts, USA, 1999.
- [25] L. Zhang, S. Malik. Cache Performance of SAT Solvers: A Case Study for Efficient Implementation of Algorithms. *SAT*, 287-298, 2003.

Asynchronous Forward-Bounding for Distributed Constraints Optimization ¹

Amir Gershman and Amnon Meisels and Roie Zivan ²

Abstract.

A new search algorithm for solving distributed constraint optimization problems (*DisCOPs*) is presented. Agents assign variables sequentially and propagate their assignments asynchronously. The asynchronous forward-bounding algorithm (*AFB*) is a distributed optimization search algorithm that keeps one consistent partial assignment at all times. Forward bounding propagates the bounds on the cost of solutions by sending copies of the partial assignment to all unassigned agents concurrently. The algorithm is described in detail and its correctness proven. Experimental evaluation of *AFB* on random *Max-DisCSPs* reveals a phase transition as the tightness of the problem increases. This effect is analogous to the phase transition of *Max-CSP* when local consistency maintenance is applied [3]. *AFB* outperforms Synchronous Branch & Bound (*SBB*) as well as the asynchronous state-of-the-art *ADOPT* algorithm, for the harder problem instances. Both asynchronous algorithms outperform *SBB* by a large factor.

1 Introduction

The Distributed Constraint Optimization Problem (*DisCOP*) is a general framework for distributed problem solving that has a wide range of applications in Multi-Agent Systems and has generated significant interest from researchers [12, 18]. Distributed constraints optimization problems (*DisCOPs*) are composed of agents, each holding its local constraints network, that are connected by constraints among variables of different agents. Agents assign values to variables and communicate with each other, attempting to generate a solution that is globally optimal with respect to the costs of constraints between agents (cf. [12, 14]).

Distributed *COPs* are an elegant model for many every day combinatorial problems that are distributed by nature. Take for example a large hospital that is composed of many wards. Each ward constructs a weekly timetable assigning its nurses to shifts. The construction of a weekly timetable involves solving a constraint optimization problem for each ward. Some of the nurses in every ward are qualified to work in the *Emergency Room*. Hospital regulations require a certain number of qualified nurses (e.g. for Emergency Room) in each shift. This imposes constraints among the timetables of different wards and generates a complex Distributed COP [16, 2].

Several distributed search algorithms for *DisCOPs* have been proposed in recent years [17, 12, 14]. The present paper proposes a new distributed search algorithm for *DisCOPs*, *Asynchronous Forward-Bounding* (*AFB*). In the *AFB* algorithm agents assign their variables

sequentially to generate a partial solution. The innovation of the proposed algorithm lies in propagating partial solutions asynchronously. Propagation of partial solutions enables asynchronous updating of bounds on their cost, and early detection of a need to backtrack, hence the algorithm's name - Asynchronous Forward Bounding.

The overall framework of the *AFB* algorithm is based on a *Branch and Bound* scheme. Agents extend a partial solution as long as the lower bound on its cost does not exceed the global bound, which is the cost of the best solution found so far. In the proposed *AFB* algorithm, (similar to the *AFC* algorithm for *DisCSPs* [11]) the state of the search process is represented by a data structure called *Current Partial Assignment* (*CPA*). The *CPA* starts empty at some initializing agent that records its assignments on it and sends it to the next agent. The cost of a *CPA* is the accumulated cost of constraints involving the value assignments it contains. Each agent which receives the *CPA*, adds assignments of its local variables such that the *CPA*'s cost will not exceed the global upper bound. If it cannot find such assignments, it backtracks by sending the *CPA* to the last assigning agent, requesting it to revise its assignment.

The innovation of the *AFB* algorithm lies in the asynchronous use of the *CPAs* by the participating agents. An agent that succeeds to extend the assignment on the *CPA* sends forward copies of the updated *CPA*, requesting all unassigned agents to compute lower bound estimations on the cost of the partial assignment (*PA*). The assigning agent will receive these estimations asynchronously over time and use them to update the lower bound of the *CPA*. Using these bounds, the assigning agent can detect if any extension of this partial assignment into a complete assignment will cause it to exceed the global upper bound, and in such cases it will backtrack. Backtracking is performed by creating a new *CPA*, a clone of the previous one, and sending it to the last assigning agent, to have that assignment replaced. A time stamp mechanism for forward checking in *DisCSPs* is used by agents to determine the most updated *CPA* and to discard obsolete *CPAs* (cf. [13]).

The concurrency of the *AFB* algorithm is achieved by the fact that forward-bounding is performed concurrently and asynchronously by all agents.

The *AFB* algorithm is described in detail in Section 3 and its correctness is proven in Section 4. The performance of *AFB* is compared to that of Synchronous Branch and Bound (*SBB*) and *ADOPT* on randomly generated *DisCOPs*. *AFB* outperforms *SBB* as well as the best asynchronous optimization algorithm *ADOPT* by a large factor, on the harder instances of random problems. This is true for two measures of distributed performance: the number of non concurrent steps of computation and the total number of messages sent (see section 5). A phase transition for problems of very high tightness is observed for *AFB*, similarly to the one that is reported for extended local consistency maintenance on *Max-CSPs* [3, 4].

¹ The research was supported by the Lynn and William Frankel Center for Computer Science, and by the Paul Ivanier Center for Robotics.

² Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, 84-105, Israel. email: {amirger,am,zivanr}@cs.bgu.ac.il

2 Distributed Constraint Optimization

A distributed constraint optimization problem (*DisCOP*) is composed of a set of **agents** A_1, \dots, A_n , each holding a set of constrained **variables**. Each variable X_i has a **domain** D_i - a set of possible values. **Constraints** (or relations) exist between variables. Each constraint involves some variables (possibly belonging to different agents), and defines a non-negative **cost** for every possible value combination of these variables. A **binary constraint** is a constraint involving only two variables. An **assignment** (or a label) is a pair including a variable, and a value from that variable's domain. A **partial assignment** (PA) is a set of assignments, in which each variable appears at most once. The **cost of a partial assignment** is computed using all constraints that involve only variables that appear in the partial assignment. Each such constraint defines some cost for the value assignments detailed in the partial assignment. All these costs are accumulated, and the sum is denoted as the cost of the partial assignment. A **full assignment** is a partial assignment that includes all the variables. A **solution** to the *DisCOP* is a full assignment of minimal cost.

A widely used special case of *DisCOPs* is to use a cost of 1 unit for each broken (unsatisfied) constraint. This type of problem is termed *Max-DisCSPs*, in accordance with *Max-CSPs* for the centralized case [3, 6]. In this paper, we will assume each agent is assigned a single variable, and use the term “agent” and “variable” interchangeably. We will assume that constraints are at most binary and the delay in delivering a message is finite. Furthermore, we assume a static final order on the agents, known to all agents participating in the search process. These assumptions are commonly used in *DisCSP* and *DisCOP* algorithms [17, 12]. For the experimental section we used *Max-DisCSPs* [3, 12]. *Max-DisCSPs* are a special class of *DisCOPs* in which all costs of binary constraints are exactly one.

3 Asynchronous Forward Bounding - AFB

In the *AFB* algorithm a single most up-to-date current partial assignment is passed among the agents. Agents assign their variables only when they hold the up-to-date *CPA*. The *CPA* is a unique message that is passed between agents, and carries the partial assignment that agents attempt to extend into a complete and optimal solution by assigning their variables on it. The *CPA* also carries the accumulated cost of constraints between all assignments it contains, as well as a unique time-stamp.

Only one agent performs an assignment on a *CPA* at a time. Copies of the *CPA* are sent forward and are concurrently processed by multiple agents. Each unassigned agent computes a lower bound on the cost of assigning a value to its variable, and sends this bound back to the agent which performed the assignment. The assigning agent uses these bounds to prune sub-spaces of the search-space which do not contain a full assignment with a cost lower than the best full assignment found so far.

More specifically, every agent that adds its assignment to the *CPA* sends forward copies of the *CPA*, in messages we term *FB_CPA*, to all agents whose assignments are not yet on the *CPA*. An Agent receiving an *FB_CPA* message computes a lower bound on the cost increment caused by adding an assignment to its variable. This estimated cost is sent back to the agent who sent the *FB_CPA* message via *FB_ESTIMATE* messages. The following two paragraphs describe how exactly this estimation is computed.

Denote by $\text{cost}((i, v), (j, u))$ the cost of assigning $A_i = v$ and $A_j = u$. For each agent A_i and each value in its domain $v \in D_i$, we denote the minimal cost of the assignment (i, v) incurred by an agent

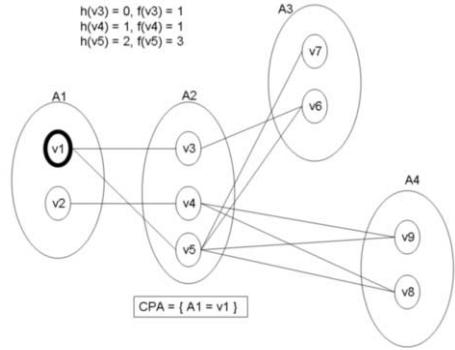


Figure 1. A simple DisCOP, demonstration

A_j by $h_j(v) = \min_{u \in D_j} (\text{cost}((i, v), (j, u)))$. We define $h(v)$, the total cost of assigning the value v , to be the sum of $h_j(v)$ over all $j > i$. Intuitively, $h(v)$ is a lower bound on the cost of constraints involving the assignment $A_i = v$ and all agents A_j such that $j > i$. Note that this bound can be computed once per agent, since it is independent of the assignments of higher priority agents.

An agent A_i , which receives an *FB_CPA* message, can compute for every $v \in D_i$ both the cost increment of assigning v as its value, i.e. the sum of the cost of conflicts v has with the assignments included in the *CPA*, and $h(v)$. The sum of these, is denoted by $f(v)$. The lowest calculated $f(v)$ among all values $v \in D_i$ is chosen to be the lower bound estimation of agent A_i .

Figure 1 presents a constraint network, in which A_1 already assigned the value v_1 and A_2, A_3, A_4 are unassigned. Let us assume that the cost of every constraint is one. The cost of v_3 will increase by one due to its constraint with the current assignment thus $f(v_3) = 1$. Since v_4 is constrained with both v_8 and v_9 , assigning this value will trigger a cost increment when A_4 performs an assignment. Therefore $h(v_4) = 1$ is an admissible lower bound of the cost of the constraints between this value and lower priority agents. Since v_4 does not conflict with assignments on the *CPA*, $f(v_4) = 1$ as well. $f(v_5) = 3$ because this assignment conflicts with the assignment on the *CPA* and in addition conflicts with all the values of the two remaining agents.

Since $h(v)$ takes into account only constraints of A_i with lower priority agents (A_j s.t. $j > i$), unassigned lower priority agents do not need to estimate their cost of constraints with A_i . Therefore, these estimations can be accumulated and summed up by the agent which initiated the forward bounding process to compute a lower bound on the cost of a complete assignment extended from the *CPA*. Asynchronous forward bounding enables agents an early detection of partial assignments that can not be extended into complete assignments with cost smaller than the known upper bound, and initiate backtracks as early as possible.

The *AFB* algorithm is run on each of the agents in the *DisCOP*. Each agent first calls the procedure *init* and then responds to messages until it receives a *TERMINATE* message. The algorithm is presented in Figure 2

3.1 Algorithm description

The algorithm starts by each agent calling **init** and then awaiting messages until termination. At first, each agent updates B to be the cost of the best full assignment found so far and since no such assignment was found, it is set to infinity (line 1). Only the first agent (A_1) creates an empty *CPA* and then begins the search process by

```

procedure init:
1.  $B \leftarrow \infty$ 
2. if ( $A_i = A_1$ )
3.   generate_CPA()
4.   assign_CPA()

```

when received (**FB_CPA**, A_j , PA)

```

5.  $f \leftarrow$  estimation based on the received  $PA$ .
6. send (FB_ESTIMATE,  $f$ ,  $PA$ ,  $A_i$ ) to  $A_j$ 

```

when received (**CPA_MSG**, PA)

```

7.  $CPA \leftarrow PA$ 
8.  $TempCPA \leftarrow PA$ 
9. if  $TempCPA$  contains an assignment to  $A_i$ , remove it
10. if ( $TempCPA.cost \geq B$ )
11.   backtrack()
12. else
13.   assign_CPA()

```

when received (**FB_ESTIMATE**, estimate, PA , A_j)

```

14. save estimate
15. if ( $CPA.cost + \text{all saved estimates} \geq B$ )
16.   assign_CPA()

```

when received (**NEW_SOLUTION**, PA)

```

17.  $B\_CPA \leftarrow PA$ 
18.  $B \leftarrow PA.cost$ 

```

procedure **assign_CPA**:

```

19. clear estimations
20. if  $CPA$  contains an assignment  $A_i = w$ , remove it
21. iterate (from last assigned value) over  $D_i$  until found
     $v \in D_i$  s.t.  $CPA.cost + f(v) < B$ 
22. if no such value exists
23.   backtrack()
24. else
25.   assign  $A_i = v$ 
26.   if  $CPA$  is a full assignment
27.     broadcast (NEW_SOLUTION,  $CPA$ )
28.      $B \leftarrow CPA.cost$ 
29.     assign_CPA()
30.   else
31.     send(CPA_MSG,  $CPA$ ) to  $A_{i+1}$ 
32.     forall  $j > i$ 
33.     send(FB_CPA,  $A_i$ ,  $CPA$ ) to  $A_j$ 

```

procedure **backtrack**:

```

34. clear estimates
35. if ( $A_i = A_1$ )
36.   broadcast(TERMINATE)
37. else
38.   send(CPA_MSG,  $CPA$ ) to  $A_{i-1}$ 

```

Figure 2. The procedures of the AFB Algorithm

calling **assign_CPA** (lines 3-4), in order to find a value assignment for its variable.

An agent receiving a CPA (when received **CPA_MSG**), first makes sure it is relevant. The time stamp mechanism suggested by [13] is used to determine the relevance of the **CPA**.

If the CPA's time-stamp reveals that it is not the most up to date

CPA, the message is discarded. In such a case, the agent processing the message has already received a message implying that an assignment of some agent which has a higher priority than itself, has been changed. When the message is not discarded, the agent saves the received PA in its local CPA variable (line 7). Then, the agent checks that the received PA (without an assignment to its own variable) does not exceed the allowed cost B (lines 8-10). If it does not exceed the bound, it tries to assign a value to its variable (or replace its existing assignment in case it has one already) by calling **assign_CPA** (line 13). If the bound is exceeded, a backtrack is initiated (line 11) and the CPA is sent to a higher priority agent, since the cost is already too high (even without an assignment to its variable).

Procedure **assign_CPA** attempts to find a value assignment, for the current agent, within the bounds of the current CPA. First, estimates related to prior assignments are cleared (line 19). Next, the agent attempts to assign every value in its domain it did not already try. If the CPA arrived without an assignment to its variable, it tries every value in its domain. Otherwise, the search for such a value is continued from the value following the last assigned value. The assigned value must be such that the sum of the cost of the CPA and the lower bound of the cost increment caused by the assignment will not exceed the upper bound B (lines 20-22). If no such value is found, then the assignment of some higher priority agent must be altered, and so backtrack is called (line 23). Otherwise, the agent assigns the selected value on the CPA.

When the agent is the last agent (A_n), a complete assignment has been reached, with an accumulated cost lower than B , and it is broadcast to all agents (line 27). This broadcast will inform the agents of the new bound for the cost of a full assignment, and cause them to update their upper bound B .

The agent holding the CPA (A_n) continues the search, by updating its bound B , and calling **assign_CPA** (line 29). The current value will not be picked by this call, since the CPA's cost with this assignment is now equal to B , and the procedure demands the cost to be lower than B . So the agent will continue the search, testing other values, and backtracking in case they do not lead to further improvement.

When the agent holding the CPA is not the last agent (line 30), the CPA is sent forward to the next unassigned agent, for additional value assignment (line 31). Concurrently, forward bounding requests (i.e. **FB_CPA** messages) are sent to all lower priority agents (lines 32-33).

An Agent receiving a forward bounding request (when received **FB_CPA**) from agent A_j , again uses the time-stamp mechanism to ignore irrelevant messages. Only if the message is relevant, then the agent computes its estimate (lower bound) of the cost incurred by the lowest cost assignment to its variable (line 5). The exact computation of this estimation was described above (it is the minimal $f(v)$ over all $v \in D_i$). This estimation is then attached to the message and sent back to the sender, as a **FB_ESTIMATE** message.

An agent receiving a bound estimation (when received **FB_ESTIMATE**) from a lower priority agent A_j (in response to a forward bounding message) ignores it if it is an estimate to an already abandoned partial assignment (identified using the time-stamp mechanism). Otherwise, it saves this estimate (line 14) and checks if this new estimate causes the current partial assignment to exceed the bound B (line 15). In such a case, the agent calls **assign_CPA** (line 16) in order to change its value assignment (or backtrack in case a valid assignment cannot be found).

The call to **backtrack** is made whenever the current agent cannot find a valid value (i.e. below the bound B). In such a case, the agent clears its saved estimates, and sends the CPA backwards to agent A_{i-1} (line 38). If the agent is the first agent (nowhere to backtrack

to), the terminate broadcast ends the search process in all agents (line 36). The algorithm then reports that the optimal solution has a cost of B , and the full assignment with such cost is B_CPA .

4 Correctness of AFB

In order to prove correctness for *AFB* two claims must be established. First, that the algorithm terminates and second that when the algorithm terminates its global upper bound B is the cost of the optimal solution. To prove termination one can show that the *AFB* algorithm never goes into an endless loop. To prove the last statement it is enough to show that the same partial assignment cannot be generated more than once.

Lemma 1 *The AFB algorithm never generates two identical CPAs.*

Assume by negation that A_i is the highest priority agent (first in the order of assignments) that generates a partial assignment *CPA* for the second time. The replacement of an assignment can only be triggered by one of two messages arriving at A_i from a lower priority agent A_j ($j > i$). Either a backtrack *CPA* message, or a *FB_ESTIMATE* message. In the first case the next assignment on the *CPA* will be generated by the procedure *assign_CPA*. Each of the values in the domain of A_i is considered exactly once. When the agent's domain is exhausted the agent backtracks and under the above assumption will never receive the same partial assignment again. If the received message is an estimate that clashes with the upper bound (e.g. the second case), a new *CPA* is generated. The generated *CPA* is a clone of the last *CPA* the agent received from a higher priority agent. Only values which were not considered on the previous *CPA* are left in its current domain. Therefore, the situation with the new *CPA* is similar to the first case.

Termination follows immediately from Lemma 1.

Next we prove that on termination, the complete assignment, corresponding to the optimal solution is in B_CPA (see Figure 2). There is only one point of termination for the *AFB* algorithm, in procedure *backtrack*. So, one needs to prove that during search no partial assignment that can lead to a solution of lower cost than B is discarded. But, this fact is immediate, because the only place in the code where values are discarded is in the third line of procedure *assign_CPA* (line 21). Within this procedure, values are discarded only when the calculated lower bound of the value being considered is higher than the current bound on the cost of a global solution. Clearly, this cannot lead to a discarding of a lower cost global solution.

One still needs to show that whenever the algorithm calls the procedure *assign_CPA*, it does not loose a potential lower cost solution. There are altogether 4 places in the algorithm, where a call to procedure *assign_CPA* is made. One is in procedure *init*, which is irrelevant. The three relevant calls are in the code performed when receiving a *CPA* or receiving a *FB_ESTIMATE*, and in the procedure *assign_CPA* itself.

The third case is trivially correct. Before calling the procedure the global bound B is updated and the corresponding complete solution is stored. Consequently, the current solution is not lost. The first two calls (see Figure 2) appear in the last lines of the procedures processing the two messages. When processing a *FB_ESTIMATE* message, the call to *assign_CPA* happens after the lower bound of the current value has been tested to exceed the global bound B (line 15). This is correct, since the current partial solution cannot be extended to a lower cost solution. The last call to *assign_CPA* occurs in the last line of processing a received *CPA* message. Clearly, this

call extends a shorter partial solution and does not discard a value of the current agent. This completes the correctness proof of the *AFB* algorithm.

5 Experimental Evaluation

All experiments were performed on a simulator in which agents are simulated by threads which communicate only through message passing. The Distributed Optimization problems used in all of the presented experiments are random *Max-DisCSPs*. *Max-DisCSP* is a subclass of *DisCOP* in which all constraint costs (weights) are equal to one [12]. The network of constraints, in each of the experiments, is generated randomly by selecting the probability p_1 of a constraint among any pair of variables and the probability p_2 , for the occurrence of a violation (a non zero cost) among two assignments of values to a constrained pair of variables. Such uniform random constraints networks of n variables, k values in each domain, a constraints density of p_1 and tightness p_2 are commonly used in experimental evaluations of CSP algorithms (cf. [15]). *Max-CSP* was used in experimental evaluations of constraint optimization problems (*COPs*) by [3, 6]. Other experimental evaluations of *DisCOPs* include graph coloring problems [12, 18], which are a subclass of *Max-DisCSP*.

In order to evaluate the performance of distributed algorithms, two independent measures of performance are commonly used - run time, in the form of non-concurrent steps of computation, and communication load, in the form of the total number of messages sent [7, 17]. We use the method described in [10] for counting non-concurrent computational steps.

AFB is compared with the *ADOPT* algorithm, which is the state-of-the-art algorithm for *DisCOPs*. Our implementation of *ADOPT* is based on [12] as well as open code published by the designer of *ADOPT*, Jay Modi.

The *ADOPT* algorithm (as described in [12]) starts by first constructing a pseudo-tree of the agents. After this initial construction, each agent informs agents below it in the tree of its current value assignment. Each agent is responsible for reporting to its parent the optimal assignment cost of the sub problem for which it is a root.

The algorithm is asynchronous, therefore each agent reports only lower and upper bounds on its sub-problem. As the sub-problem is explored, these bounds are refined until eventually, the lower bound and upper bound become equal. At that point, the sub-problem is optimally solved [12].

In order to avoid thrashing through redundant assignment replacing, *ADOPT* uses a “threshold splitting” mechanism. Our implementation uses the same heuristic for *threshold splitting* used in the open code implementation (the “*splitThresholdOneChild*” heuristic). The construction of the *pseudo tree* is generated also according to the original implementation, using the well known *DFS* algorithm.

To help reduce the number of messages, we improved *ADOPT*'s implementation, having each agent read its entire mailbox, process all awaiting messages and only then send the messages needed. A similar improvement was done for the *ABT* algorithm [17, 19, 1]. In our experimental evaluation, we considered the processing of all these messages as a single computational step.

Synchronous Branch and Bound (*SBB*) is a simple algorithm, that simulates centralized Branch & Bound in a distributed environment [17]. A single *CPA* which carries the search state, and is passed between agents is all that is needed.

Figure 3 presents the average run-time in number of computation steps, of the three algorithms *AFB*, *ADOPT* and *SBB* on *Max-DisCSPs* with $n = 10$ agents, domain size $k = 10$, and a constraint density of $p_1 = 0.4$. The value of p_2 was varied between 0.4 and

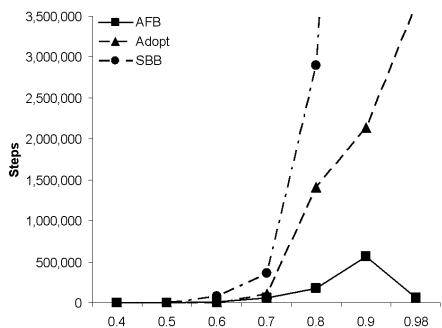


Figure 3. Total Non-concurrent computational steps by AFB, ADOPT and SBB on Max-DisCSP.

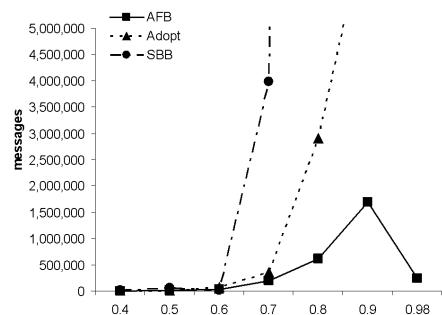


Figure 4. Total number of messages sent by AFB, ADOPT and SBB.

0.98, to cover all ranges of problem difficulty [15].

It is clear in Figure 3 that as the tightness becomes larger and the problem harder, *AFB* outperforms *ADOPT* which outperforms *SBB*. For tightness values that are higher than 0.9 *AFB* demonstrates a phase transition, while *ADOPT*'s and *SBB*'s run-time keeps growing. The behavior of *AFB* is very similar to that of lookahead algorithms on centralized *Max-CSPs* [3, 4, 6]. The run time of *ADOPT* ad *SBB* increases exponentially fast, so that on the very extreme values of p_2 it simply did not terminate in a reasonable time and their execution was terminated. Figure 4 shows the total number of messages sent by the three algorithms. *AFB* clearly outperforms *ADOPT* which outperforms *SBB* in this measure.

6 Conclusion

The present paper proposes a *DisCOP* algorithm that performs Forward-Bounding asynchronously. Extended consistency maintenance procedures are performed concurrently. The results presented in Section 5 demonstrate the importance of consistency maintenance for distributed search. The concurrent form of extended lookahead (forward bounding) prevents an exponential growth in run-time for the tighter problem instances. This is similar to the centralized case of *Max-CSPs* [3, 6]. The run-time performance of *AFB* is far better than *ADOPT* on tight distributed *Max-DisCSPs*. The advantage of *AFB* over *ADOPT* in overall network load is even more pronounced (Figure 4).

Recent studies of centralized constraints optimization problems have shown that the use of extended consistency maintenance procedures improves the performance of simple Branch and Bound [4, 6]. In particular, the hardest problem instances (with high p_2 value) show a phase transition, in which run-time decreases [3, 4, 5, 6].

Previous studies of distributed *COPs* have shown many advantages of the *ADOPT* algorithm over *SBB*, which is the naive algo-

rithm for solving these problems [12]. Previous experimental evaluations of *ADOPT* measured its scalability, by increasing the number of variables/agents ([12, 9, 8]). The behavior of *ADOPT* on the hardest (tightest) *Max-DisCSP* instances was not explored. The experiments in the present study measure the algorithm's performance over varying problem difficulty levels. The exponential growth of concurrent run-time of *ADOPT* for very hard and tight problem instances is therefore, only apparent in the present study.

REFERENCES

- [1] C. Bessiere, A. Maestre, I. Brito, and P. Meseguer, ‘Asynchronous backtracking without adding links: a new member in the abt family’, *Artificial Intelligence*, **161:1-2**, 7–24, (January 2005).
- [2] E. Kaplansky and A. Meisels, ‘Distributed personnel scheduling: Negotiation among scheduling agents’, *Annals of Operations Research*, (2005).
- [3] J. Larrosa and P. Meseguer, ‘Phase transition in max-csp’, in *Proc. ECAI-96*, Budapest, (1996).
- [4] J. Larrosa, P. Meseguer, and T. Schiex, ‘Maintaining reversible dac for max-csp.’, *Artificial Intelligence*, **107**, 149–163, (1999).
- [5] J. Larrosa and T. Scheix, ‘In the quest for the best form of local consistency for weighted csp’, in *Proc. IJCAI-2003*, Acapulco, (2003).
- [6] J. Larrosa and T. Schiex, ‘Solving weighted csp by maintaining arc consistency.’, *Artificial Intelligence*, **159**, 1–26, (2004).
- [7] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Series, 1997.
- [8] T. Maheswaran, M. Tambe, E. Bowring, J.P. Pearce, and P. Varakantham, ‘Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling’, in *Proc. AAMAS-2004*, New York, (July 2004).
- [9] Roger Mailer and Victor Lesser, ‘Solving distributed constraint optimization problems using cooperative mediation’, in *Proc. AAMAS-2004*, pp. 438–445, (July 2004).
- [10] A. Meisels, I. Razgon, E. Kaplansky, and R. Zivan, ‘Comparing performance of distributed constraints processing algorithms’, in *Proc. AAMAS-2002 Workshop on Distributed Constraint Reasoning DCR*, pp. 86–93, Bologna, (July 2002).
- [11] A. Meisels and R. Zivan, ‘Asynchronous forward-checking for distributed csp’s’, in *Frontiers in Artificial Intelligence and Applications*, ed., W. Zhang. IOS Press, (2003).
- [12] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, ‘Adopt: asynchronous distributed constraints optimization with quality guarantees’, *Artificial Intelligence*, **161:1-2**, 149–180, (January 2005).
- [13] T. Nguyen, D. Sam-Houd, and B. Faltings, ‘Dynamic distributed back-jumping’, in *Proc. 5th workshop on distributed constraints reasoning DCR-04*, Toronto, (September 2004).
- [14] A. Petcu and B. Faltings, ‘A value ordering heuristic for distributed resource allocation’, in *Proc. CSCLP04, Lausanne, Switzerland*, <http://liawww.epfl.ch/Publications/Archive/Petcu2004.pdf>, (2004).
- [15] P. Prosser, ‘An empirical study of phase transitions in binary constraint satisfaction problems’, *Artificial Intelligence*, **81**, 81–109, (1996).
- [16] G. Solotorevsky, E. Gudes, and A. Meisels, ‘Modeling and solving distributed constraint satisfaction problems (dcsp’s)’, in *Constraint Processing-96*, pp. 561–2, New Hampshire, (October 1996).
- [17] M. Yokoo, ‘Algorithms for distributed constraint satisfaction problems: A review’, *Autonomous Agents & Multi-Agent Sys.*, **3**, 198–212, (2000).
- [18] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg, ‘Distributed stochastic search and distributed breakout: properties, comparison and applications to constraints optimization problems in sensor networks’, *Artificial Intelligence*, **161:1-2**, 55–88, (January 2005).
- [19] R. Zivan and A. Meisels, ‘Synchronous vs asynchronous search on dcsp’s’, in *Proc. 1st European Workshop on Multi Agent System, EUMAS*, Oxford, (December 2003).

Distributed Log-based Reconciliation

Yek Loong Chong¹ and Youssef Hamadi²

Abstract. Computer Supported Cooperative Work (CSCW) defines software tools and technology to support groups of people working together on a project, often at different sites [5]. In this work, we present four distributed algorithms for log-based reconciliation, an important NP-hard problem occurring in CSCW. Our methods remove the classical drawbacks of centralized systems like single point of failure, performance bottleneck and loss of autonomy. The problem is formalized using the Distributed Constraint Satisfaction paradigm (DisCSP). In the worst case, the message passing complexity of our methods range from $O(p^2)$ to $O(2^p)$ in a system of p nodes. Experimental results confirm our theoretical analysis and allow us to establish quality and efficiency trade-off for each method.

Keywords: Distributed Constraint Satisfaction

1 INTRODUCTION

Optimistic replication is a technology of choice for Computer Supported Cooperative Work (CSCW) where users collaborate asynchronously to achieve a common goal. The major assumption in optimistic replication systems is that conflicting operations are exceptional [10] and can be resolved through a process called reconciliation. In these systems, users share a set of common data objects through replication and update their local replicas independently. These isolated updates can introduce divergence in the states of the shared objects which are occasionally repaired through reconciliation. In *log-based* reconciliation, local updates are recorded in logs containing the isolated operations. These logs represent the input of the reconciliation process. The output or *schedule*, which is an ordered collection of non-conflicting operations, is computed by combining the isolated operations in some order. The schedule when executed against a previously consistent state results in a new consistent state. However, a set of isolated updates may contain conflicting operations and combining these operations may violate some invariant. One way of preserving correctness is to exclude conflicting operations from the schedule. An efficient reconciliation algorithm minimizes these exclusions.

A major criticism of current optimistic systems is that reconciliation is centralized, which exhibits classical drawbacks like single point of failure, performance bottleneck and loss of autonomy. Drawbacks aside, due to various factors like organizational boundaries and security/privacy, centralization may indeed be impossible or undesirable. What is needed is a method for making decision locally and incrementally, without having to accumulate all information at one site. In this work, we present a set of distributed algorithms for log-based reconciliation that avoid the previous drawbacks and preserve privacy.

As demonstrated in [9], the semantic of several CSCW systems can be represented using constraints. These systems can be modelled and solved in the Constraints formalism [7]. In this work, we provide the first distributed algorithms for distributed log-based reconciliation. These algorithms are defined in the Distributed Constraint Satisfaction Problem paradigm (DisCSP) [12].

This paper is organised as follows. We first provide a high level overview of the different domains in section 2. Section 3 presents the modelling of log-based reconciliation as a distributed constraint satisfaction problem. Our algorithms are presented in section 4 and their theoretical analysis is given in section 5. Before the general conclusion provided in the last section, section 6 presents detailed experimental results.

2 BACKGROUND

2.1 Constraint-based Optimistic Replication

In order to maintain consistency in a log-based reconciliation system, different updates to the same object must be executed in the same sequence at every replica. This can be complicated due to the varying latencies involved in accessing replicas that are distributed across a wide-area network. This problem is further exacerbated in mobile computing where the availability of replicas is also variable. Many optimistic systems like Bayou [2] and IceCube [9], circumvent this problem by using a centralised commit protocol (also known as the primary-based commit). In these systems, a distinguished site or *primary* is elected by the system for committing tentative operations. The order in which tentative operations are committed at the primary is deemed to be the authoritative order. Tentative operations must be executed at all replicas in the order specified by the authoritative sequence. IceCube is an optimistic replication system that supports multiple applications and data types using a concept called constraints between operations. The input is made by a set of logs coming from p users or sites $\{(a_{i1}, \dots, a_{ij}) | 1 \leq i \leq p\}$ where a_{ij} represents the j^{th} action for user i . These logs are individually consistent i.e., operations recorded in a log do not violate any correctness criterion at the originating site. The schedule is an ordered collection of non-conflicting operations (a_1, \dots, a_k) which represents the largest subset of users' operations consistent with the following conditions :

1. *before* or *temporal* constraint : for any a_i, a_j in the schedule, a_i comes before a_j (not necessarily immediately before) : $a_i \rightarrow a_j$
2. *must have* or *dependency* constraint : for any a_i in the schedule, a_j must also be in the schedule : $a_i \triangleright a_j$

Operations which are not part of the final schedule are deemed conflicting and aborted by the system. The semantics of a large set of applications can be represented through the combination of these low level constraints. For instance, the fact that a method f_m of a software object o needs to start with the call to an initialisation method f_i can be encoded into, $o.f_i \rightarrow o.f_m, o.f_m \triangleright o.f_i$.

¹ University of Cambridge, Computer Laboratory, 15 J J Thomson Avenue, Cambridge CB3 0FD, United Kingdom, ylc21@hermes.cam.ac.uk

² Microsoft Research Ltd., 7 J J Thomson Avenue, Cambridge CB3 0FB, United Kingdom, youssef@microsoft.com

A consistent schedule must respect any dependency relation and avoid oriented cycles of temporal constraints. The previous problem restricted to *before* constraints can be reduced to the search of the largest acyclic network of actions which is NP-hard.

2.2 Distributed Constraint Satisfaction

DisCSP is a general paradigm usually deployed in multi-agent applications where the global outcome depends on the joint decisions of autonomous agents. Examples of such applications are distributed planning and distributed sensor networks management [3]. Informally, a DisCSP is represented by a set of variables, each of which is associated with a domain of values, and a set of constraints that restrict combinations of values between variables. The variables are partitioned amongst a set of agents, such that each agent owns a proper subset of the variables. The task is for each agent to assign a value to each variable it owns without violating the constraints.

Solving a DisCSP is equivalent to finding an assignment of values to variables such that all the constraints are satisfied.

3 MODELLING THE DISTRIBUTED LOG-BASED RECONCILIATION PROBLEM

Efficiently solving the log-based reconciliation problem involves the finding of the largest set of non conflicting actions. This optimisation problem could theoretically be solved with a distributed optimisation procedure like [11]. This algorithm can find an optimal solution or a solution within a user-specified distance from the optimal. Unfortunately, it is too space intensive for our purpose and does not scale beyond a few dozens of variables. Therefore, we decided to apply a pragmatic strategy which computes locally optimal solutions (optimal within the scope of each agent) and globally combines them into a consistent distributed solution. The drawback of this approach comes from the loss of global optimality, but the benefit is efficiency. Moreover, final schedules are reasonably good since local decision making can take into account peers decisions. We now detail our full DisCSP model in the following.

Assuming that there are n tentative operations from p sites or agents, we use two constrained variables x_i and s_i to represent each tentative operation a_i . Therefore, a distributed reconciliation problem formulated as a DisCSP is a quadruplet (X, D, C, A) where :

1. X is a set of $2n$ variables $x_1, \dots, x_n, s_1, \dots, s_n$.
2. D is the set of domains $D_{x_1}, \dots, D_{x_n}, D_{s_1}, \dots, D_{s_n}$ of possible values for the variables $x_1, \dots, x_n, s_1, \dots, s_n$ respectively, where:
 - $D_{x_i} = [0..1]$, the value 1 is set when the action (or operation) a_i is selected for inclusion in the schedule.
 - $D_{s_i} = [-n^2 .. n^2]$, each value represents a possible position for action a_i in the schedule³.
3. C is the set of predicates defining the constraints (see section 2.1) between the operations:
 - (a) $(x_i == 1) \implies (s_i < s_j)$ to express any $a_i \rightarrow a_j$ relation.
 - (b) $x_j \geq x_i$ to express any $a_i \triangleright a_j$ relation.
4. A is a partitioning of X amongst the set of p autonomous agents P_1, \dots, P_p defined by $f : X \rightarrow A$ such that $f(x_i) = P_k \Leftrightarrow f(s_i) = P_k$.

³ Intuitively, n discrete values would be adequate to represent the positions of n operations. However, this is not the case with our distributed algorithms (see sec. 4).

Finally, each P_k uses a local branch-and-bound search which maximizes, $\sum_{a_i \in P_k} x_i$. It represents the maximizing of the size of the local schedule. Search is performed over the x_i with bound consistency on the s_i . At the end of the distributed exploration, any selected operation a_i (s.t., $x_i = 1$) can use the lower bound of the associated s_i as a valid ordering position in the schedule.

4 DISTRIBUTED ALGORITHMS FOR LOG-BASED RECONCILIATION

4.1 Distributed ordering of the agents

Our solution synthesis algorithms are totally asynchronous. However, they require an ordering $>_o$ between related agents (two agents are related if they share some constraint). We decided to use a general method, Distributed Agent Ordering (DisAO) which is classically used to order a network of agents [8]. DisAO computes a distributed acyclic graph of agents where related agents are always strictly ordered by $>_o$. It can use any domain-dependent heuristic to guide its reasoning.

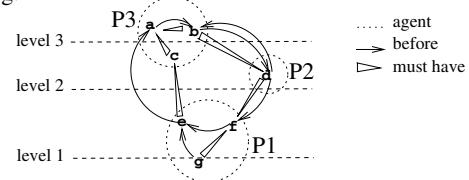


Figure 1. Distributed Ordering of the agents

Figure 1 presents a small reconciliation problem ordered with DisAO. Each agent is presented with internal actions and constraints. It shows that $P_3 >_o P_2 >_o P_1$. Agent P_3 has two *Children*, P_1 and P_2 . P_2 has one parent and one child while at the bottom, P_1 has two parents. These *Parents* and *Children* sets represent primaries with higher (resp. lower) priorities.

4.2 Distributed Reconciliation Algorithm

Our first procedure is presented in Algorithm 1. It represents a powerful problem-specific extension of ABT [12]. For each agent, the input is made by the previously defined *Parents* and *Children* sets. The output is a locally optimal schedule consistent with both intra- and inter-constraints. Solution synthesis begins with the simultaneous initialisation of the nodes. *localView* the data structure representing parents' solutions is initialized and a search for s , a locally optimal solution is performed. The method *maximize* runs a local branch-and-bound which maximizes the size of the local schedule. Before this first call to the local solver [6], the domains D_{s_i} are updated thanks to a first call to *resolveDomain*. This method sets the domain of each local variable s_i to $D_{s_i} = [0..n_k - 1]$, where n_k represents the number of local actions. This represents the initial basis or referential of each agent knowing that n_k values are enough to schedule n_k local operations at that stage. Afterwards, s is propagated to the *Children* acquaintances through *infoVal* messages. Interactions can then start through an event loop. First, agents check for termination. Such messages are addressed by some entity able to detect global stabilization of the system (see [1]). Next, *infoVal* messages are processed. If the current solution s appears to be compatible with the new information, the processing stops. Note that the current solution may be compatible but suboptimal w.r.t. incoming information.

Now, if some inconsistency is detected, a new locally optimal solution must be found. This process starts with a *rewriting* of the incoming solution and a computation of the new ranges of the s_i variables. Intuitively, we only require n_k discrete values to represent the

position of n_k local operations. However, agent P_k searching for a new local solution must consider its *Parents* solutions. Clearly, some local operation a_i could be scheduled between, before or after ancestors' operations. The rewriting of the incoming solution allows the scheduling of local operations between ancestor ones. This process is presented in algorithm 2. It takes an ordered incoming solution S_{in} and rewrites it into S'_{in} , the same solution where actions are shifted to allow the potential scheduling of n_k local actions in between. Finally, to authorize the scheduling of the local operations before or after ancestors ones, the local s_i variables are set to $D_{s_i} = [\min(\{s_i | s_i \in \text{localView}\}) - n_k .. \max(\{s_i | s_i \in \text{localView}\}) + n_k]$. The call to the *resolveDomain* function sets the previous bounds.

Algorithm 1: Distributed Reconciliation

Input: Parents, Children sets;
Output: A local schedule consistent with intra- and inter-constraints;
begin

```

localView:=∅; resolveDomains(localView);
s:=maximize(local problem); send('infoVal', s, Children);
end:=false;
while (!end) do
  msg:=getMsg();
  if (msg.type = "terminate") then
    end:=true;
  if (msg.type = "infoVal") then
    if (!localView.compatible(s)) then
      msg.content := rewriteSolution(msg.content);
      localView.add(msg.content);
      resolveDomains(localView);
      s:=maximize(local problem);
      if (s) then
        send('infoVal', s, Children);
      else
        conflicts:=∅;
        ng:=computeNogoods(conflicts);
        send('backtrack',ng,conflicts);
        localView.remove(msg.content);
        s:=maximize(local problem);
        send('infoVal', s, Children);

    if (msg.type = "backtrack") then
      storeNogoods(msg.content);
      s:=maximize(local problem);
      if (s) then
        send('infoVal', s, Children);
      else
        conflicts:=∅;
        ng:=computeNogoods(conflicts);
        send('backtrack',ng,conflicts);

  end

```

Algorithm 2: rewriteSolution

Input: $S_{in} = \{(x_0, s_0), \dots, (x_k, s_k)\}$, sorted on s_i ;
Output: $S'_{in} = \{(x_0, s'_0), \dots, (x_k, s'_k)\}$;
begin

```

s'_0 := s_0; i:=1;
while (i ≤ k) do
  s'_i := s'_{i-1} + s_i - s_{i-1} + n_k + 1;
  i++;

```

If a new solution exists, it is propagated to related children. If such a solution cannot be found, a backtrack step is started. This implies first the detection of the parents involved in the conflict, second the detection of the external actions causing the conflict. A call to *computeNogoods* returns these two sets. Then, a *backtrack* message is addressed to related parents. At that point, the agent can remove

the current content from the local view and look for a new satisfying solution. Looking for a new solution instead of waiting for the consequence of the *backtrack* message has a positive impact on robustness in case of transient failures. Then, an agent considers any backtracking request. First, the incoming nogoods are stored. Subsequent searches will have to respect these new constraints. The agent looks for a new locally optimal solution s compatible with local constraints, local view and stored nogoods. If a consistent solution is found, it is propagated. If not, a new backtrack is initiated as before. Since, in log-based reconciliation, an empty schedule is perceived as consistent, termination can only happen when agents are waiting for incoming messages and when no such messages transit in the communication network. In this situation each agent has a local solution consistent with its *Parents*.

4.3 Example

Figure 2 shows from left to right a step by step execution of the algorithm on the small reconciliation example of figure 1. Each snapshot includes the local state of each action a_i in the form (x_i, s_i) .

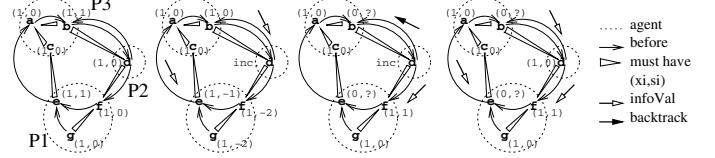


Figure 2. A complete example of distributed reconciliation

Processing starts at top left with the concurrent computation of a locally optimal solution for each sub-problem. At that step, all the actions are included in the schedule with respect to intra-constraints. Then in the second snapshot, we assume that P_2 and P_1 are processing incoming *infoVal* messages upcoming from P_3 . P_1 finds a new local solution consistent with P_3 , while P_2 detects an inconsistency. This inconsistency comes from the cycle of *before* constraints between actions b and d combined to the dependency constraints $b \triangleright d$. Indeed, removing action d in order to avoid the inconsistency upcoming from the cycle is not compatible with the dependency constraint. In the third snapshot, the nogood ($x_b \neq 1$) is addressed to P_3 in a *backtrack* message. At the same time, P_1 processes the *infoVal* message upcoming from P_2 . It finds a new local solution excluding e but compatible with both P_3 and P_2 . In the last snapshot, P_3 finds a new solution which respects its recorded nogood and propagates it towards P_1 and P_2 . Then P_2 can find for the first time a local solution compatible with P_3 . It addresses it towards P_1 . Finally P_1 successively processes two *infoval* messages and finds them compatible with its current local solution. At that stage termination is detected. The final quality of the distributed schedule is 5 (actions) while a globally optimal solution would have a quality of 6 (excluding action b). The overall message cost is 7.

4.4 Backtrack-free Distributed Reconciliation

The previous example shows that our algorithm can easily handle inconsistencies upcoming from distributed cycles of *before*. More generally,

Property 4.1 *Distributed cycles of temporal constraints can always be broken by an agent of lowest priority.*

The proof is direct if we consider that DisAO totally orders related agents. Therefore, there is always a unique agent of lowest priority for each distributed cycle of *before* which can always break

the cycle by excluding some local action. For example, in figure 2, (a, b, d, f, e) can always be broken in P_1 by removing either f or e .

Property 4.2 *Distributed backtracking is the result of the scheduling of some operation a which depends on some external operation b ($a \triangleright b$) managed by an agent of lower priority.*

From 4.1, we know that inter-temporal constraints cannot force a system to backtrack. Therefore, backtracking comes from inter-dependency constraints. Indeed, if b cannot be scheduled in the child, a has to be withdrawn. This is performed thanks to a distributed backtracking message, e.g., actions b and d in figure 2.

We can easily specialize Algorithm 1 into a new backtrack-free method. In the new procedure, a node proactively excludes actions whose scheduling depends on some child's decision. From property 4.2, the new method is backtrack-free.

When we apply this algorithm to the problem of figure 2, P_3 and P_2 proactively remove b and d . At the end of the resolution, the overall quality is 5 (P_1 can keep three actions) while the message passing cost is just 3.

4.5 Optimal Orderings

The backtrack-free algorithm proactively excludes actions which rely on children decisions. Therefore, an immediate objective is to avoid these costly situations by prioritizing the agents in a way which circumvents these *badly oriented* dependencies. In a general case, this is not always possible since a system can contain distributed cycles of dependencies. However, these cycles may intersect, and breaking as many of them at their intersection minimizes the number of badly oriented constraints and consequently, the number of forced exclusions. Ideally, we would do this by computing a minimal cycle-cut set in the oriented dual graph of dependency constraints.

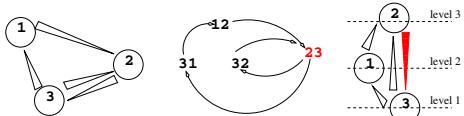


Figure 3. Distributed cycles of dependencies, dual graph and optimal ordering

Figure 3 illustrates this. On the left part we have a system with three agents linked through two cycles of dependencies. An optimal ordering will decide $P_2 >_o P_3$. Indeed, this decision breaks the two cycles in the dual graph of dependencies. The final ordering, $P_2 >_o P_1 >_o P_3$, has just one badly oriented constraint, which is coloured in red. It is presented on the right. In the middle, the dual graph is presented with its cut set coloured in red. Unfortunately, the problem of finding a minimal cycle-cut set is NP-hard [4] and we can just find ways to approximate its solution. We do this with DisAO (see sec. 4.1) which takes as input the following new heuristic function:

1. Compute locally inconsistent orderings, $P_j >_o P_i >_o P_k$ (where $P_j \triangleright P_i \triangleright P_k$).
2. Propagate these orderings in the neighbourhood.
3. Collect inconsistent orderings, prune local combinations of orderings by using collected information.

The first step detects situations where some action in P_j relies on some action managed by P_i while at the same time some possibly different action in P_i relies on some action managed by P_k . These situations must be avoided. The last step assumes that a node manages all the possible local orderings involving itself and its neighbours. It uses incoming inconsistencies to filter these local orderings. When it receives some inconsistency $P_j \triangleright P_i \triangleright P_k$, it prunes all

the orderings assuming $P_j \triangleright P_i$ or $P_i \triangleright P_k$. The message passing complexity of this calculation is $O(p^2)$, see [8].

The application of the previous heuristic to our main example, gives the optimal ordering presented in figure 4. We can observe that, action e relies on action c , which means that the top priority agent depends on the decision of a child. With this ordering, our distributed reconciliation algorithm uses 4 messages and find an optimal solution which schedules 6 actions (b 's removal breaks the two distributed cycles of *before*). The backtrack-free method requires 4 messages and schedules 5 actions.

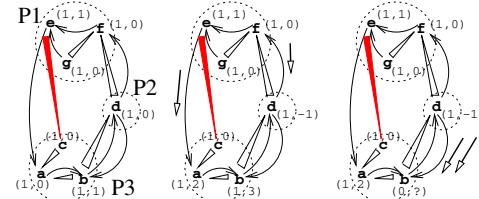


Figure 4. An optimal ordering

If we include the previous pre-processing step which minimizes dependencies towards children decisions, we finally have four methods for distributed log-based reconciliation:

1. The general distributed reconciliation : *backtracking*.
2. The backtrack-free algorithm : *backtrack-free*.
3. The general distributed reconciliation which uses the previous pre-processing : *pre-proc+backtracking*.
4. The backtrack-free algorithm which uses the previous pre-processing : *pre-proc+backtrack-free*.

5 THEORETICAL ANALYSIS

5.1 Distributed Reconciliation

To demonstrate *correctness*, we must first show that rewritten solutions are equivalent to original ones. Because of space considerations, we do not detail the proof, which involves the demonstration that dependency constraints are not affected by the rewriting; that original inter-space is preserved and, that added inter-space does not go against the original consistency of S_{in} . Second, we must show that the dynamic narrowing of the local domains D_{s_i} is correct. By definition, algorithm 2 adds at least n_k scheduling positions between scheduled actions upcoming from parents. Therefore, it gives enough space to schedule the n_k local actions between any pair of external actions. Finally, n_k values are reserved before (resp. after) the lowest (resp. highest) scheduling position in the local view (call to *resolve-Domain*). Therefore, enough space is provided to schedule local operations before, between or after parents' ones. Third, we must show that when the system stops, computed solutions are consistent. The proof is related to the correctness of [1] which is used to detect situations where, 1.) the communication network is empty and 2.) agents are waiting for incoming messages. To show *completeness*, we need to show that if there is a solution, our algorithm can reach it and that the algorithm eventually *terminates*. Here, because of space considerations, we refer readers to [12].

Computationally, the worst case is made of a fully connected network of p agents individually managing a single action. Therefore, any DisAO ordering uses p levels. Insolubility is proven after an exhaustive exploration of the distributed search space. Since this exploration is done on the x_i variables, the search space size is 2^p . Hence, in the worst case, time and message passing complexities are $O(2^p)$.

5.2 Backtrack-free Distributed Reconciliation

Completeness (w.r.t., pessimistic decision making), correctness and termination are obvious with this algorithm. However, the complexity must be detailed. As, with the previous worst case, each agent proactively removes its local action to avoid incoming backtracks. The top level agent sends $p - 1$ *infoVal* messages while the others send a number of messages related to their position in the distributed ordering. Since this ordering has p levels, time and message passing complexities are $O(p^2)$.

6 EXPERIMENTAL RESULTS

Experiments were conducted over a set of randomly generated inputs. We simulated a system of $p = 10$ agents with a total number of $n = 100$ actions. Each agent P_k was allocated exactly $n_k = 10$ actions. Two tightness parameters p_b and p_{mh} were used to define respectively the number of *before* and *must have* constraints. Each parameter represents a percentage over the set of all possible constraints. Since hard internal sub-problems impact the overall quality but do not raise high interaction levels, we decided to restrict the hardness of each sub-problem to respectively 9 *before* and 5 *must have* constraints. The amount of inter-constraints was defined as, $tightness \times (\sum_{k=1}^p n_k \times (n - n_k))$ with both tightness parameters ranging from 0.1% to 1.5%. These values range over the space of realistic log-based reconciliation problems which appear to be under-constrained [9]. Indeed when the previous parameters are pushed they rapidly meet overconstrained regions with empty solutions (see [7] for full landscape analysis). Each system had from 18 to 270 inter-constraints either *before* or *must have*. Overall, our experiments used 200 variables and a maximum of 410 constraints. They represent the largest experiments in DisCSPs as far as our knowledge goes. For each combination of parameters we present average values over 20 random instances.

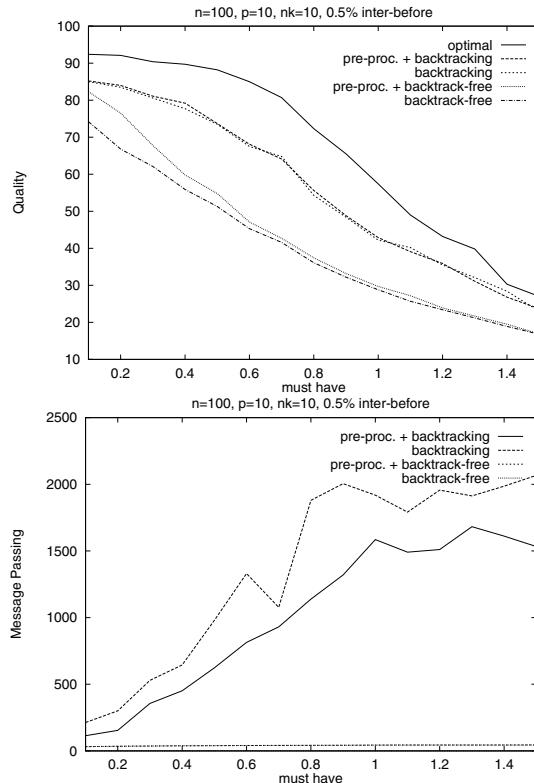


Figure 5. Quality and message passing results

Figure 5 present quality and message passing results at $p_b = 0.5\%$, i.e., 45 inter-*before* constraints. When we consider the quality, we can see that *backtracking* and *pre-proc backtracking* are very close. Note that optimal quality is provided (top curve). On the other-hand, the *backtrack-free* method really benefits from the pre-processing. It organizes the system in a way which reduces the number of pessimistic choices. The biggest win appears at $p_{mh} = 0.2$ with an improvement of 14.5%. Message passing presents a different dynamic. The pre-processing clearly improves the *backtracking* algorithm; up to 48% of the messages are saved at $p_{mh} = 0.2$. As expected, it has no impact on the message cost of the *backtrack-free* algorithm.

7 CONCLUSION

We have proposed a set of distributed algorithms which perform distributed log-based reconciliation, an important problem in CSCW. Our first method iteratively corrects locally optimal solutions through distributed backtracking. Its worst case time and message passing complexities are $O(2^p)$. Our second method is backtrack-free. It applies a pessimistic decision making which is built on a fine analysis of backtrack prone situations. Its worst case time and message passing complexities are bounded by $O(p^2)$. Moreover, we showed that in a general case, distributed cycles prohibit the complete suppression of badly oriented dependencies, i.e., a non-pessimistic procedure cannot be backtrack-free. Furthermore, we showed that it was possible to reduce backtrack prone situations; a perfect reduction being equivalent to the NP-hard computation of a minimal cycle-cut set in the dual graph of inter-dependency constraints. To approximate a perfect reduction, we have also presented a specific agent ordering heuristic. It was experimentally shown that by using this heuristic, 1) the backtrack-free procedure improves the overall quality of its solutions (up to 14.5%), 2) the general backtracking algorithm decreases its message passing consumption (up to 48%).

References

- [1] K. M. Chandy and L. Lamport, ‘Distributed snapshots: Determining global states of distributed systems’, *TOCS*, 3(1), 63–75, (Feb 1985).
- [2] A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch, ‘The bayou architecture: Support for data sharing among mobile users’, in *Proc. IEEE W. on Mobile Computing Systems & Applications*, pp. 2–7, (1994).
- [3] S. Fitzpatrick and L. Meertens, ‘Scalable, anytime constraint optimization through iterated, peer-to-peer interaction in sparsely-connected networks’, in *Proc. Sixth Biennial World Conf. on Integrated Design & Process Technology (IDPT 2002)*, (2002).
- [4] M. R Garey and D. S Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, chapter A2, W.H. Freeman and Co., San Francisco, 1979.
- [5] I. Greif, *Computer-supported cooperative work: a book of readings*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [6] Y. Hamadi, ‘Disolver : A Distributed Constraint Solver’, Technical Report MSR-TR-2003-91, Microsoft Research, (Dec 2003).
- [7] Y. Hamadi, ‘Cycle-cut decomposition and log-based reconciliation’, in *ICAPS, W. Connecting Planning Theory with Practice*, pp. 30–35, (2004).
- [8] Y. Hamadi, C. Bessière, and J. Quinqueton, ‘Backtracking in distributed constraint networks’, in *ECAI*, pp. 219–223, (Aug 1998).
- [9] A. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel, ‘The Ice-Cube approach to the reconciliation of divergent replicas’, in *20th ACM PODC Symp.*, (2001).
- [10] H. T. Kung and J. T. Robinson, ‘On optimistic methods for concurrency control’, *ACM Transactions on Database Systems*, 6, 213–226, (1981).
- [11] M. Tambe P. Modi, W.-M. Shen and M. Yokoo, ‘An asynchronous complete method for distributed constraint optimization’, in *AAMAS*, (2003).
- [12] M. Yokoo, *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*, Springer, 2001.

Extracting MUCs from Constraint Networks

Fred Hemery and Christophe Lecoutre and Lakhdar Sais and Frédéric Boussemart¹

Abstract. We address the problem of extracting Minimal Unsatisfiable Cores (MUCs) from constraint networks. This computationally hard problem has a practical interest in many application domains such as configuration, planning, diagnosis, etc. Indeed, identifying one or several disjoint MUCs can help circumscribe different sources of inconsistency in order to repair a system. In this paper, we propose an original approach that involves performing successive runs of a complete backtracking search, using constraint weighting, in order to surround an inconsistent part of a network, before identifying all *transition* constraints belonging to a MUC using a dichotomic process. We show the effectiveness of this approach, both theoretically and experimentally.

1 Introduction

A constraint network is said to be minimal unsatisfiable if and only if it is unsatisfiable and deleting an arbitrary constraint makes it satisfiable. Deciding whether a set of constraints is minimal unsatisfiable is well known to be DP-Complete [20]. It can be reduced to the SAT-UNSAT problem: given two CNF formulas ϕ and ψ , is ϕ satisfiable and ψ unsatisfiable? DP corresponds to the second level of the Boolean hierarchy. A problem in this class can be considered as the difference between two NP-problems.

On the practical side, when inconsistency is encountered, circumscribing the conflicting parts of a system can help the user understand, explain, diagnose and restore consistency. To illustrate the importance of the problem addressed in this paper, one can mention the well-known Radio Link Frequency Assignment Problem (RLFAP) which is often used as a benchmark in the CSP (Constraint Satisfaction Problem) community. This problem involves assigning frequencies to a set of radio links defined between pairs of transmitters in order to avoid interferences. To this end, one looks for a solution that minimizes the number of used frequencies. Circumscribing the unfeasible subnetwork areas (of minimal size) can help find new positions of the transmitters.

In the case of Boolean constraints (formula in Conjunctive Normal Form), finding minimal unsatisfiable sub-formula is an active research area. Tractable classes have been exhibited. Most of them are based on the deficiency of the formula (i.e. difference between the number of clauses and variables) [4, 7]. Also, recent advances in satisfiability checking has allowed successful extensions of SAT solvers for handling such a hard computational problem [3, 23, 15, 19].

In the context of constraint satisfaction, there is a significant amount of work dealing with the identification of conflict sets of constraints. Such sets, which can be built by recording explanations during search, are usually used to perform different forms of intelligent backtracking (e.g. [22, 8, 13]). However, there are only a few works

really dedicated to the extraction of MUCs from constraint networks. An approach for the diagnosis of over-constrained networks has been proposed in [1] and a method to find all MUCs from a given set of constraints is presented in [9, 5]. This method corresponds to an exhaustive exploration of a so-called CS-tree, but is limited by the combinatorial explosion in the number of subsets of constraints. Finally, a divide and conquer approach has been proposed in [12] in order to extract from an over-constrained problem an explanation (or relaxation) using preferences given by the user.

In this paper, we propose an original approach to extract a MUC from a given constraint network. This approach consists of two stages. The first one exploits the conflict-directed variable ordering heuristic *dom/wdeg* [2] in order to surround (and then extract) an unsatisfiable core by performing successive complete runs of a backtracking search algorithm. Search is restarted, while preserving constraint weighting from one run to the next one, until the size of the proved unsatisfiable core cannot be made smaller. Then, using a total order on the constraints based on their current weights, and following the principle introduced in [6], the second stage allows iteratively identifying the constraints of a MUC. Compared to *constructive* [6] and *destructive* [1] approaches which are respectively $O(e.k_e)$ and $\theta(e)$, the *dichotomic* approach that we propose is $O(\log(e).k_e)$. Here, the complexity corresponds to the worst-case number of calls to the backtracking search algorithm, e denotes the number of constraints of the given constraint network and k_e denotes the number of constraints of the extracted MUC. We also relate this complexity with the one obtained by Junker [12].

The paper is organized as follows. First, we introduce some technical background. Then, we present the two stages of our approach: extracting an unsatisfiable core by exploiting constraint weighting and extracting a minimal core by identifying so-called *transition* constraints. Next, related work is discussed. Finally, before concluding, we present the results of an experimentation that we have conducted.

2 Technical Background

A Constraint Network (CN) P is a pair $(\mathcal{X}, \mathcal{C})$ where \mathcal{X} is a finite set of n variables and \mathcal{C} a finite set of e constraints. Each variable $X \in \mathcal{X}$ has an associated domain, denoted $dom(X)$, which contains the set of values allowed for X . Each constraint $C \in \mathcal{C}$ involves a subset of variables of \mathcal{X} , called scope, and has an associated relation, denoted $rel(C)$, which contains the set of tuples allowed for the variables of its scope. For any subset $S \subseteq \mathcal{C}$ of constraints of P , $P^{\uparrow S}$ will denote the constraint network obtained from P by removing all constraints of S and $P_{\downarrow S}$ will be equivalent to $P^{\uparrow(\mathcal{C}-S)}$.

A solution to a CN is an assignment of values to all the variables such that all the constraints are satisfied. A CN is said to be satisfiable iff it admits at least one solution. The Constraint Satisfaction Problem (CSP) is the NP-complete task of determining whether a given

¹ CRIL-CNRS FRE 2499, rue de l'université, SP 16, 62307 Lens cedex, France. email: {hemery, lecoutre, sais, boussemart}@cril.univ-artois.fr

CN is satisfiable. A CSP instance is then defined by a CN, and solving it involves either finding one (or more) solution or determining its unsatisfiability. To solve a CSP instance, a depth-first search algorithm with backtracking can be applied, where at each step of the search, a variable assignment is performed followed by a filtering process called constraint propagation. Usually, constraint propagation algorithms, which are based on some constraint network properties such as arc consistency, remove some values which can not occur in any solution. The algorithm that maintains arc consistency during search is called MAC. An unsatisfiable core corresponds to an unsatisfiable subnetwork of a CN.

Definition 1. Let $P = (\mathcal{X}, \mathcal{C})$, $P' = (\mathcal{X}', \mathcal{C}')$ be two CNs. P' is an unsatisfiable core of P iff P' is unsatisfiable, $\mathcal{X}' \subseteq \mathcal{X} \wedge \mathcal{C}' \subseteq \mathcal{C}$.

Different unsatisfiable cores of a given CN may exist. Those which do not contain any proper unsatisfiable core are said to be minimal.

Definition 2. Let $P = (\mathcal{X}, \mathcal{C})$ be a CN and $P' = (\mathcal{X}', \mathcal{C}')$ an unsatisfiable core of P . P' is a Minimal Unsatisfiable Core (MUC) of P iff it does not exist any unsatisfiable core P'' of P' s.t. $P'' \neq P'$.

To show the minimality of an unsatisfiable core, one can just check the satisfiability of any CN obtained when removing one constraint.

3 Extracting Unsatisfiable Cores

First, following the idea given in [1], we introduce an approach that allows removing some constraints (while preserving unsatisfiability). Then, we refine this approach by exploiting constraint weighting and (complete) restarts.

3.1 A Proof-based Approach

When the unsatisfiability of a CSP instance is proved by a filtering search algorithm, one can automatically extract a core that is guaranteed to be unsatisfiable. Indeed, it suffices to keep track of all the constraints that have been involved in the proof of unsatisfiability, that is to say, any constraint that has been used during search to remove, by propagation, at least one value in the domain of a variable. This principle was mentioned in [1] and can be related to the concept of implication graph used in SAT (e.g. see [17, 23]).

Let us examine how it works with MAC which maintains arc consistency by exploiting, for instance, an algorithm such as AC3 [16]. It involves successive revisions of arcs (pairs composed of a constraint and of a variable) in order to remove the values that are no more consistent with the current state. At the heart of the solver is then the function depicted in Algorithm 1. All values of the domain of the given variable that are not currently supported by the given constraint are removed (lines 2 to 4).

By introducing a data structure, denoted *active*, that allows associating a Boolean with each constraint, we are then in a position to extract an unsatisfiable core. The function *pcore* depicted in Algorithm 2 allows such an extraction. Initially, all Booleans are set to *false* (line 1). Then, the MAC solver is called (line 2), what involves successive revisions. Hence, whenever a revision is effective, the Boolean associated with the constraint is set to *true* (line 5 of Algorithm 1). Finally, the function returns (line 3) the CN obtained from P by removing any constraint C such that $active[C]$ is *false*. It is important to remark that the network returned by *pcore* is guaranteed to be unsatisfiable but not necessarily minimal.

Algorithm 1 *revise*(C : Constraint, X : Variable) : Boolean

```

1: domainSize  $\leftarrow |dom(X)|$ 
2: for each  $a \in dom(X)$  do
3:   if seekSupport( $C, X, a$ ) = false then
4:     remove  $a$  from dom( $X$ )
5:     active[ $C$ ]  $\leftarrow true$ 
6: if dom( $X$ ) =  $\emptyset$  then
7:   wght[ $C$ ]  $\leftarrow wght[C] + 1$  // used by some heuristics
8: return domainSize  $\neq |dom(X)|$ 
```

Algorithm 2 *pcore*($P = (\mathcal{V}, \mathcal{C})$: CN) : CN

```

1: active[ $C$ ]  $\leftarrow false$ ,  $\forall C \in \mathcal{C}$ 
2: MAC( $P$ )
3: return  $P^{\uparrow \{C \in \mathcal{C} | active[C] = false\}}$ 
```

3.2 A Conflict-based Approach

Even if the proof-based approach is an elegant approach, we have no idea about its practical efficiency. In other words, we cannot predict the size of the unsatisfiable core extracted by *pcore*. It is clear that the smallest the size is, the most efficient the approach is. Actually, as illustrated below, exploiting a conflict-directed variable ordering heuristic in order to push up an unsatisfiable core by performing successive runs makes the proof-based approach quite effective.

Heuristic *dom/wdeg* In [2], it is proposed to associate a counter, denoted *wght*[C], with any constraint C of the problem. These counters are used as constraint weighting. Whenever a constraint is shown to be unsatisfied (during the constraint propagation process), its weight is incremented by 1 (see line 7 of Algorithm 1). The weighted degree of a variable X is then defined as the sum of the weights of the constraints involving X and at least another uninstantiated variable. The conflict-directed heuristic *dom/wdeg* [2] involves selecting first the variable with the smallest ratio current domain size to current weighted degree. As search progresses, the weight of hard constraints become more and more important and this particularly helps the heuristic to select variables appearing in the hard part of the network. This heuristic has been shown to be quite efficient [2, 14, 10].

Illustrative Example Using *dom/wdeg* allows efficiently proving the unsatisfiability of many instances. However, in order to obtain a proof of unsatisfiability of moderate size, one has to be aware that it is important to perform successive runs by restarting search several times. As an illustration, let us consider the problem of putting some queens and some knights on a chessboard as described in [2]. The instance with 6 queens and 3 knights involves 9 variables and 36 constraints and is unsatisfiable. In fact, we know that the subproblem corresponding to the 3 knights, and involving 3 variables and 3 constraints, is unsatisfiable. In a first phase, solving this instance with MAC-*dom/wdeg* (i.e. MAC combined with the *dom/wdeg* variable ordering heuristic) yields a proof of unsatisfiability integrating all constraints of the instance (that is to say, all Boolean *active* have been set to *true*). However, solving again the same instance, using current weighting of the constraints as obtained after the first run, yields a new proof of unsatisfiability integrating only 9 constraints. An additional run furnishes the same result.

Figure 1 illustrates the evolution of such proofs of unsatisfiability. CNs are represented by constraint graphs where vertices correspond to variables and edges to binary constraints. Note that constraints ir-

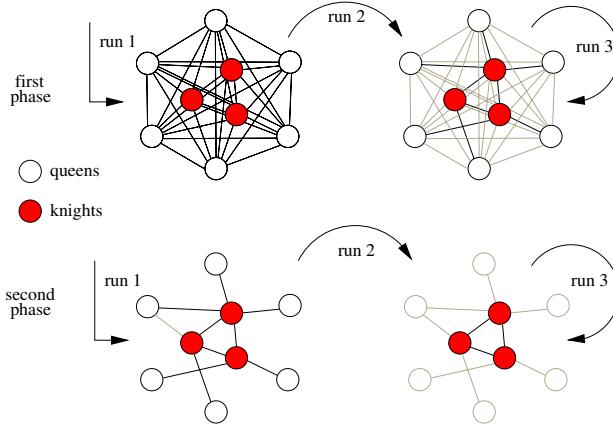


Figure 1. Evolution of the proof of unsatisfiability

relevant to unsatisfiability after one run are represented by dashed edges. Then, it is possible to refine the extraction after removing all the constraints which are not involved in the detected core, i.e., proof of unsatisfiability. Indeed, in a second phase, we obtain an unsatisfiable core that corresponds to the knights subproblem.

Exploiting Conflict-directed Heuristics As illustrated above, performing several runs of a MAC solver may be useful to surround an unsatisfiable core provided that a conflict-directed heuristic such as *dom/wdeg* is used. This approach is depicted by Algorithm 3. Initially (line 1), the weight of all constraints is set to 1. Then, iteratively, MAC-*dom/wdeg* is run (line 6) and the number of constraints found in the unsatisfiable core detected by the current run is counted (line 7). The iteration stops when the size of the current unsatisfiable core is greater than or equal to the size of the previous one. Remember that from one run to the next one, the *wght* counters are preserved, which allows potentially concentrating the search to a smaller and smaller unsatisfiable core. Note that we can easily generalize this algorithm in order to perform several phases as mentioned in the illustration.

Algorithm 3 $wcore(P = (\mathcal{V}, \mathcal{C}) : CN) : CN$

```

1:  $wght[C] \leftarrow 1, \forall C \in \mathcal{C}$ 
2:  $cnt_{aft} \leftarrow +\infty$ 
3: repeat
4:    $active[C] \leftarrow false \forall C \in \mathcal{C}$ 
5:    $cnt_{bef} \leftarrow cnt_{aft}$ 
6:   MAC-dom/wdeg( $P$ )
7:    $cnt_{aft} \leftarrow |\{C \in \mathcal{C} | active[C]\}|$ 
8: until  $cnt_{aft} \geq cnt_{bef}$ 
9: return  $P \upharpoonright \{C \in \mathcal{C} | active[C] = false\}$ 

```

4 Extracting Minimal Unsatisfiable Cores

It is clear that the unsatisfiable core that can be extracted by using the function *wcore* is not guaranteed to be minimal. In order to find a minimal core, it is necessary to iteratively identify the constraints that are involved in it. More precisely, we know that, given an unsatisfiable CN P and a total ordering of the constraints (to simplify, we shall consider the natural lexicographic order C_1, C_2, \dots, C_e of the constraints), there exists a constraint C_i such that $P \upharpoonright \{C_1, \dots, C_{i-1}\}$ is satisfiable and $P \upharpoonright \{C_1, \dots, C_i\}$ is unsatisfiable². This constraint which

² We shall assume that no constraint C exists in P such that $\text{rel}(C) = \emptyset$.

clearly belongs to a minimal core of P will be called the *transition constraint* of P (according to the given ordering). Note also that any constraint C_j with $j > i$ can be safely removed.

4.1 Identifying the Transition Constraint

It is possible to identify the *transition constraint* of an unsatisfiable CN P by using a constructive approach, a destructive approach or a dichotomic one. Below, $\text{MAC}(P)$ returns *SAT* (reps. *UNSAT*) iff P is satisfiable (resp. unsatisfiable), and the parameter $k (< |\mathcal{C}|)$ indicates the number of transition constraints previously identified (the first k constraints of the current network). It will be meaningful later, but initially, just consider $k = 0$.

Constructive Approach The principle is to successively add the constraints of the given network until the current network becomes unsatisfiable. This approach is analog to the one introduced in [6] and is depicted by Algorithm 4.

Algorithm 4 $csTransition(P = (\mathcal{V}, \mathcal{C}) : CN, k : int) : Constraint$

```

1: for  $i$  increasingly varying from  $k + 1$  to  $|\mathcal{C}|$  do
2:   if  $\text{MAC}(P \upharpoonright \{C_1, \dots, C_i\}) = \text{UNSAT}$  then return  $C_i$ 

```

Destructive Approach The principle is to successively remove the constraints of the given network until the current network becomes satisfiable (note² that i can never reach 1). This approach has been introduced in [1] and is depicted by Algorithm 5.

Algorithm 5 $dsTransition(P = (\mathcal{V}, \mathcal{C}) : CN, k : int) : Constraint$

```

1: for  $i$  decreasingly varying from  $|\mathcal{C}|$  to  $k + 1$  do
2:   if  $\text{MAC}(P \upharpoonright \{C_1, \dots, C_{i-1}\}) = \text{SAT}$  then return  $C_i$ 

```

Dichotomic Approach Finally, it is possible to use a dichotomic search in order to find the transition constraint. This approach is depicted by Algorithm 6. At each step of the search, we know that the transition constraint of P belongs to $\{C_{min}, \dots, C_{max}\}$.

Algorithm 6 $dcTransition(P = (\mathcal{V}, \mathcal{C}) : CN, k : int) : Constraint$

```

1:  $min \leftarrow k + 1 ; max \leftarrow |\mathcal{C}|$ 
2: while  $min \neq max$  do
3:    $center \leftarrow (min + max)/2$ 
4:   if  $\text{MAC}(P \upharpoonright \{C_1, \dots, C_{center}\}) = \text{SAT}$  then  $min \leftarrow center + 1$ 
5:   else  $max \leftarrow center$ 
6: return  $C_{min}$ 

```

4.2 Extracting the Minimal Core

Using one of the functions described above allows finding the transition constraint C_i of an unsatisfiable network P , that is to say one element that belongs to a minimal core of P . To get a second element, we apply the same function on a new CN P' which is obtained from P by removing all constraints C_j such that $j > i$ (since unsatisfiability is preserved) and considering a new order of the constraints such that C_i is considered as the smallest element (a natural order can be preserved by simply renaming constraints). This process can be repeated until all constraints of the current network correspond to transition constraints that have been successively found. The principle of this iterative process has been described in [6, 11, 21]. It is

depicted by Algorithm 7 which returns a MUC from the given network (just consider one of the tree approaches by replacing xx with cs, ds or dc). Note that we have to determine if the last constraint belongs to the MUC (lines 7 and 8).

Algorithm 7 $xxMUC(P = (\mathcal{V}, \mathcal{C}) : CN) : CN$

```

1:  $P' \leftarrow P ; k \leftarrow 0$ 
2: while  $k < |\mathcal{C}'| - 1$  do
3:    $C_i \leftarrow xxTransition(P', k)$ 
4:    $k \leftarrow k + 1$ 
5:    $P' \leftarrow P'^{\uparrow\{C_j | j > i\}}$ 
6:   in  $P'$ ,  $tmp \leftarrow C_i$ ,  $C_{j+1} \leftarrow C_j$  for  $1 \leq j < i$ ,  $C_1 \leftarrow tmp$ 
7: if  $MAC(P'^{\uparrow\{C_{|\mathcal{C}'|}\}}) = UNSAT$  then return  $P'^{\uparrow\{C_{|\mathcal{C}'|}\}}$ 
8: else return  $P'$ 

```

The following proposition (whose proof is omitted) suggests that the dichotomic approach should be more efficient than the two other ones.

Proposition 1. *Let $P = (\mathcal{X}, \mathcal{C})$ be an unsatisfiable CN. The worst-case number of calls to MAC is $O(e \cdot k_e)$ for csMUC(P), $\theta(e)$ for dsMUC(P) and $O(\log(e) \cdot k_e)$ for dcMUC(P). Here, $e = |\mathcal{C}|$ and k_e is the number of constraints of the extracted MUC.*

To be more precise, we obtain a worst-case number of calls to MAC by dcMUC bounded by $\log_2(e) \cdot (k_e + 1)$ (+1 since we have to prove that the k_e transition constraints form a MUC). It can be compared with QuickXplain whose worst-case complexity is $2k_e \cdot \log_2(e/k_e) + 2k_e$ [12]. In the worst-case, dcMUC is better than QuickXplain when $(\log_2(k_e) - 1) \cdot 2k_e / (k_e - 1) < \log_2(e)$, that is to say, when the size of the extracted core is rather small. This condition holds for all the instances that have been tested in our experimentation. Also, considering the illustration given in [12] with $e = 2^{20}$ and $k_e = 2^3$, we obtain 288 calls with QuickXplain against 180 calls with dcMUC.

Finally, for efficiency reasons, it is really important to use, in practice, a conflict-based approach (function *wcore*) before extracting a MUC (function *xxMUC*). This will be shown in Section 6. The methods that we consider are then (given a constraint network P):

- CS which corresponds to call csMUC(*wcore*(P))
- DS which corresponds to call dsMUC(*wcore*(P))
- DC which corresponds to call dcMUC(*wcore*(P))

Even if it does not explicitly appear above, we will consider that all constraints are renamed, before calling xxMUC, in such a way that the lexicographic order corresponds to the decreasing order of the current weights of the constraints. It allows to improve the methods by limiting the number of runs performed. Note also that we can arbitrarily bound the number of calls to MAC by *wcore* in order to have Proposition 1 hold for CS, DS and DC. In practice, we have observed that the number of calls to MAC by *wcore* is always low.

5 Related Work

On the one hand, research in extraction of unsatisfiable cores of constraint networks is rather limited. Bakker et al. [1] have proposed a method to extract a MUC (in the context of Model-Based Diagnosis). This method essentially corresponds to call dsMUC(*pcore*(P)). Our approach can then be seen as a refinement³ of theirs since we propose

³ It is also important to note that weights introduced in [1] correspond to static preferences given by the user.

a dichotomic approach (dcMUC) and a conflict-based preliminary stage (*wcore*) whose practical importance will be shown in Section 6. Some other works [11, 21] concern the identification of minimal II conflict-sets where II denotes a propagation operator. Roughly speaking, while extracting a MUC is an activity which is global to the network, extracting a II conflict-set is an activity limited to a branch of the search tree. As a consequence, in order to keep some incrementality of the propagation process, the proposed algorithms in [11, 21] involves (at least, partially) a constructive schema. The (new) method QuickXplain [12] exploits a divide and conquer approach (which exploits a dichotomic process) and whose complexity has been discussed in Section 4.2. A similar approach, called XC1, has been proposed in [18] in a more general context.

On the other hand, research in unsatisfiable cores for propositional satisfiability (SAT) is quite active. Bruni and Sassano [3] have proposed an “adaptive core search” to recover a small unsatisfiable sub-formula. Clause hardness are evaluated thanks to an history search analysis. By selecting a fixed percentage of hard clauses, the current unsatisfiable core is expended or contracted until the core becomes unsatisfiable. In [23], using a resolution proof based approach, the Zchaff solver is extended for approximating an unsatisfiable core (i.e. the returned unsatisfiable core is not guaranteed to be minimal). Finally, Lynce and Marques-Silva [15] have proposed a model that computes a *minimum* unsatisfiable core (i.e. the smallest unsatisfiable core in the number of clauses).

6 Experiments

In order to show the practical interest of the approach described in this paper, we have conducted an experimentation, using for each run MAC-dom/wdeg, on a PC Pentium IV 2.4GHz 512Mo under Linux. Performances have been measured in terms of the number of runs (#runs) and the cpu time in seconds (cpu). We also indicate the number of constraints (#C) (and, sometimes, of variables (#V)) of instances and extracted cores.

Our experimentation has been performed wrt some random and real-world instances. The random instances correspond to the unsatisfiable instances of two sets, denoted *ehi-85-297* and *ehi-90-315*, containing 100 easy random instances embedding a small unsatisfiable part. The real-world instances correspond to the two archives RLFAP and FAPP. The selected unsatisfiable instances of the Radio Link Frequency Assignment Problem (RLFAP) came from the CELAR (Centre electronique de l’armement) while the instances of the Frequency Assignment with Polarization Problem (FAPP) came from the ROADEF’2001 challenge. Most of these instances were used as benchmarks⁴ for the first CSP solver competition.

Instance	<i>pcore</i>		<i>wcore</i>	
	cpu	#C	cpu	#C
<i>qk-25-25-5-mul</i> (#C = 435)	100.1	427	107.7	32
<i>ehi-85-297-0</i> (#C = 4, 094)	2.93	3, 734	3.01	226
<i>graph-14-f28</i> (#C = 4, 638)	4.23	3, 412	4.69	503

Table 1. Cost (cpu) and Size (#C) of cores extracted by *pcore* and *wcore*

First, we have studied the practical interest of calling *wcore* instead of *pcore*. Table 1 indicates the size of the cores extracted by both methods wrt some instances. One can observe that it is really worth using *wcore* since the size of the extracted core can be very small and the additional cost of performing successive runs is not penalizing (the cost for *wcore* is given for all performed runs). It is

⁴ <http://cpai.ucc.ie/05/Benchmarks.html>

illustrated with one queens-knights instance, one random 3-SAT instance and one RLFAP instance. Note that, for some other instances, the gap between the two methods is negligible.

Then, we have compared DC with CS and DS. On random EHI instances (mean costs are given in the first part of Table 2), the difference between DS and DC is not very important. It can be explained by the fact that the extracted core returned by *wdeg* is already small (it has been observed on all EHI instances but not on all tested real-world instances). However, on RLFAP and FAPP instances (see second and third parts of Table 2), DC clearly outperforms CS and DS. Both, in terms of cpu and number of runs, DC is about 10 times more efficient than CS and DS. We have obtained this kind of results on about half the RLFAP and FAPP instances that we have tested.

Instance	Method	cpu	#runs	MUC	
				#V	#C
EHI instances (100 instances per series)					
<i>ehi</i> – 85 – 297	CS	263	1284	19	32
#V = 297	DS	62	157	23	37
#C ≈ 4100	DC	45	153	19	32
<i>ehi</i> – 90 – 315	CS	266	1294	19	32
#V = 315	DS	64	156	23	36
#C ≈ 4370	DC	46	154	19	32
RLFAP instances					
<i>graph13-w1</i>	CS	303	704	4	6
#V = 916	DS	257	338	4	6
#C = 1479	DC	39	55	4	6
<i>scen02-f25</i>	CS	145	588	10	15
#V = 200	DS	130	311	12	28
#C = 1235	DC	21	67	10	15
<i>scen09-w1-f3</i>	CS	468	576	6	8
#V = 680	DS	533	480	6	8
#C = 1138	DC	38	48	6	8
FAPP instances					
<i>fapp03-300-5</i>	CS	469	484	3	3
#V = 300	DS	333	290	3	3
#C = 2326	DC	57	37	3	3
<i>fapp06-500-1</i>	CS	394	393	3	3
#V = 500	DS	306	195	3	3
#C = 3478	DC	48	35	3	3
<i>fapp10-900-1</i>	CS	1206	688	4	4
#V = 900	DS	743	365	4	4
#C = 6071	DC	119	46	4	4

Table 2. Extracting a MUC from EHI, RLFAP and FAPP instances

Finally, we have used the three methods in order to iteratively remove, until satisfiability is reached, disjoint MUCs (i.e. MUCs that do not share any constraint) from two RLFAP instances. It takes about 5 minutes to reach this goal with DC. Remark that there are about 40 and 130 constraints involved in the set of disjoint MUCs extracted from *graph05* and *scen11-f10*, respectively. It represents about 3.5% of the set of constraints of each instance.

Instance	Method	cpu	#runs	#MUCs
<i>graph05</i>	CS	5,175	8,825	5
#V = 200	DS	1,996	2,010	5
#C = 1134	DC	330	526	5
<i>scen11-f10</i>	CS	1,491	3,840	5
#V = 680	DS	3,040	2,452	5
#C = 4103	DC	263	562	5

Table 3. Extracting successive MUCS from RLFAP instances

7 Conclusion

In this paper, we have presented a new approach, denoted DC, that allows extracting MUCs from constraint networks. The originality of this approach is that it exploits the recent heuristic *dom/wdeg* in order to surround an unsatisfiable core by performing successive complete runs of MAC, and a dichotomic search in order to identify the successive transition constraints belonging to a MUC. We have introduced both theoretical and practical arguments to support our

approach. In particular, the worst-case number of calls to MAC of DC is bounded by $\log_2(e) \cdot (k_e + 1)$, and DC has appeared to be quite efficient with respect to real-world instances taken from the RLFAP and FAPP archives. Indeed, a MUC has been extracted from most of these instances in less than 1 minute. In comparison, it is worth mentioning that most of the instances that have been experimented in this paper can not be solved, in a reasonable amount of time, when using standard variable ordering heuristics (see [2, 14]).

Acknowledgments

This paper has been supported by the CNRS, the “programme CO-COA de la Région Nord/Pas-de-Calais” and by the “IUT de Lens”.

REFERENCES

- [1] R.R. Baker, F. Dikker, F. Tempelman, and P.M. Wognum, ‘Diagnosing and solving over-determined constraint satisfaction problems’, in *Proceedings of IJCAI’93*, pp. 276–281, (1993).
- [2] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, ‘Boosting systematic search by weighting constraints’, in *Proceedings of ECAI’04*, pp. 146–150, (2004).
- [3] R. Bruni and A. Sassano, ‘Detecting minimally unsatisfiable subformulae in unsatisfiable SAT instances by means of adaptative core search’, in *Proceedings of SAT’00*, (2000).
- [4] H. Kleine Buning, ‘On subclasses of minimal unsatisfiable formulas’, *Discrete Applied Mathematics*, **107**(1-3), 83–98, (2000).
- [5] M. Garcia de la Banda, P.J. Stuckey, and J. Wazny, ‘Finding all minimal unsatisfiable subsets’, in *Proceedings of PPDP’03*, (2003).
- [6] J.L. de Siqueira and J.F. Puget, ‘Explanation-based generalisation of failures’, in *Proceedings of ECAI’88*, pp. 339–344, (1988).
- [7] H. Fleischner, O. Kullmann, and S. Szeider, ‘Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference’, *Theoretical Computer Science*, **289**, 503–516, (2002).
- [8] M. Ginsberg, ‘Dynamic backtracking’, *Artificial Intelligence*, **1**, 25–46, (1993).
- [9] B. Han and S-J. Lee, ‘Deriving minimal conflict sets by cs-trees with mark set in diagnosis from first principles’, *IEEE Transactions on Systems, Man and Cybernetics*, **29**(2), 281–286, (1999).
- [10] T. Hulubei and B. O’Sullivan, ‘Search heuristics and heavy-tailed behaviour’, in *Proceedings of CP’05*, pp. 328–342, (2005).
- [11] U. Junker, ‘QuickXplain: conflict detection for arbitrary constraint propagation algorithms’, in *Proceedings of IJCAI’01 Workshop on modelling and solving problems with constraints*, pp. 75–82, (2001).
- [12] U. Junker, ‘QuickXplain: preferred explanations and relaxations for over-constrained problems’, in *Proc. of AAAI’04*, pp. 167–172, (2004).
- [13] N. Jussien and V. Barichard, ‘The palm system: explanation-based constraint programming’, in *Proc. of TRICS’00*, pp. 118–133, (2000).
- [14] C. Lecoutre, F. Boussemart, and F. Hemery, ‘Backjump-based techniques vs conflict-directed heuristics’, *ICTAI’04*, pp. 549–557, (2004).
- [15] I. Lynce and J.P. Marques-Silva, ‘On computing minimum unsatisfiable cores’, in *Proceedings of SAT’04*, (2004).
- [16] A.K. Mackworth, ‘Consistency in networks of relations’, *Artificial Intelligence*, **8**(1), 99–118, (1977).
- [17] J.P. Marques-Silva and K.A. Sakallah, ‘Conflict analysis in search algorithms for propositional satisfiability’, Technical Report RT/4/96, IN-ESC, Lisboa, Portugal, (1996).
- [18] J. Mauss and M. Tatar, ‘Computing minimal conflicts for rich constraint languages’, in *Proceedings of ECAI’02*, pp. 151–155, (2002).
- [19] Y. Oh, M.N. Mneimneh, Z.S. Andraus, K.A. Sakallah, and I.L. Markov, ‘AMUSE: A minimally-unsatisfiable subformula extractor’, in *Proceedings of DAC’04*, pp. 518–523, (2004).
- [20] C.H. Papadimitriou and D. Wolfe, ‘The complexity of facets resolved’, *Journal of Computer and System Sciences*, **37**, 2–13, (1988).
- [21] T. Petit, C. Bessière, and J.C. Régin, ‘A general conflict-set based framework for partial constraint satisfaction’, in *Proceedings of SOFT’03 workshop held with CP’03*, (2003).
- [22] P. Prosser, ‘Hybrid algorithms for the constraint satisfaction problems’, *Computational Intelligence*, **9**(3), 268–299, (1993).
- [23] L. Zhang and S. Malik, ‘Extracting small unsatisfiable cores from unsatisfiable boolean formulas’, in *Proceedings of SAT’03*, (2003).

Preference-based Inconsistency Proving: When the Failure of the Best Is Sufficient

Ulrich Junker¹

Abstract. Inconsistency proving of CSPs is typically achieved by a combination of systematic search and arc consistency, which can both be characterized as resolution. However, it is well-known that there are cases where resolution produces exponential contradiction proofs, although proofs of polynomial size exist. For this reason, we will use optimization methods to reduce the proof size globally by 1. decomposing the original unsatisfiability problem into a conjunction of satisfiable subproblems and by 2. finding an ordering that separates the solution spaces of the subproblems. This principle allows Operation Research methods to prove the inconsistency of overconstrained linear programs even if domains are infinite. We exploit the principle for testing the satisfiability of global user requirements in product configuration problems.

1 Introduction

Many applications of constraint programming combine a generic constraint model with specific user requirements, which easily leads to overconstrained constraint satisfaction problems (CSP). For example, scheduling problems have no solution if due dates are too tight. And it may not be possible to configure a product that satisfies all user wishes, while respecting a limited budget.

The inconsistency of such a problem can be proved by a solver that maintains arc consistency (MAC) during a systematic and exhaustive tree search. MAC eliminates locally inconsistent variable-value assignments in each search node. Search then picks a still undecided variable-value assignment and makes it true in one child node and false in the other one. If all possible values of a variable have been eliminated in a search node, then the problem of this search node is arc inconsistent. The inconsistency is proved if each leave of the search tree is arc inconsistent. We can characterize the inconsistency proof by labelling each node of the search tree by the sequence of variable-value eliminations that have been effectuated in this node. The size of the proof (i.e. the number of all eliminations summed over the whole tree) is a good measure for the effectiveness of the MAC-based prover and allows us to identify problematic cases:

1. *Polynomial proof size*: arc consistency is a complete method for proving the inconsistency of tree-like constraint graphs. Although arc consistency is polynomial, it can lead to too much overhead if the variable domains and the relations of the constraints are large. This can happen in configuration problems when the complete product catalogue of a database is mapped to a CSP.
2. *Exponential proof size*: the well-known resolution principle has an exponential proof complexity as demonstrated by examples such as the pigeon-hole problem. MAC limited to binary constraints is a form of resolution and cannot do better.

3. *Infinite proof size*: the Tarski-like fixpoint definition of arc consistency can be generalized for constraints on variables with infinite domains. In this case, arc consistency as well as bound consistency can lead to an infinite number of value removals. As example, consider two integer variables x and y that are greater than 1 and the constraints $x \geq 2 \cdot y$ and $y \geq 2 \cdot x$.

Can we obtain shorter proofs if we use other proof principles? Indeed, many customized constraint propagation algorithms internally exploit dominance principles and solve optimization subproblems to test satisfiability. This applies to numerical constraints such as $x + y \leq z$. We determine the smallest value for z under $x, y \in [40, 80]$ when checking the consistency of the upper bound of $z \in [20, 60]$. The smallest value 80 for z is obtained for $x = 40$ and $y = 40$. Since it exceeds the upper bound of 60, any other value for x and y will also lead to a value for z that exceeds 60. We can thus reduce the proof size by checking only the smallest values for x and y .

Linear programming solvers proceed in a similar way when proving the unsatisfiability of overconstrained linear programs. They relax some of the constraints and replace them by an optimization objective. They use this objective to determine a best solution of the remaining constraints. If this best solution violates a relaxed constraint, then no solution exists. This approach works even if domains are infinite. Even global constraints such as constraints of all-difference of n variables solve a maximal flow optimization subproblem to find inconsistency proofs of polynomial size for pigeonhole-like problems.

In contrast to arc consistency (AC), those methods do not inspect each possible domain value. Indeed they decompose the original problem into a conjunction of two satisfiable subproblems and they determine an ordering on the possible assignments that permits a separation of the solution spaces of the two subproblems. It is then sufficient to show that the best value of one problem is worse than the worst value of the other problem. Usually one order is determined by exploiting some monotonicity property of some of the constraints.

In this paper, we give a general formulation of this separation principle based on arbitrary partial orders (Sections 3 and 4). We no longer require that the order can be expressed in form of a ranking or utility function. We thus gain in terms of flexibility for relaxing constraints. For example, we can use Pareto-dominance for relaxing bound constraints on multiple criteria. We can use CP-nets [2] to relax conditional constraints. In Section 5, we apply this theoretical framework to a practical case, namely the configuration of multiple product components under global resource constraints (such as the sum of the prices of all devices of my home cinema must not exceed my budget). Although AC is sufficient in this case, it leads to a high overhead. PrefAC [1] uses preferences to reduce this effort for arc consistent CSPs, but will converge with AC on arc inconsistent CSPs. However, we can generate preferences from global resource

¹ ILOG S.A., France, email: ujunker@ilog.fr

constraints and reduce the effort to those solutions of the remaining constraints that are the best ones w.r.t. those preferences. Thus we can significantly reduce the size of inconsistency proofs (in particular if there is a single best solution). The approach also promises an effective incorporation of databases into CSPs, in particular if the subproblem consists of a single, but large table constraint that models a product catalogue. Recent work on preference databases fits in neatly and allows the calculation of the Pareto-frontier of such a catalogue (cf. e.g. [4]). The separation principle can also be used to generate comprehensive explanations of failure for those kind of problems since the results of the optimization subproblems provide useful information for understanding the nature of the failure.

2 Constraints and Preferences

Throughout this paper, we consider a finite set of variables \mathcal{X} where each variable $x \in \mathcal{X}$ has a domain $D(x)$ (e.g. the set of integers between 0 and 1000). Each value $v \in D(x)$ defines a possible value assignment $x = v$ to x . A set that contains exactly one of those value assignments for each variable in \mathcal{X} and that contains no other elements is called an *assignment* to \mathcal{X} . The set of all assignments to \mathcal{X} is called the *problem space* of \mathcal{X} and we denote it by $\mathcal{S}(\mathcal{X})$. Given an assignment σ to \mathcal{X} we can project it to a subset Y of the variables be defining $\sigma[Y] := \{(x = v) \in \sigma \mid x \in Y\}$.

We can restrict the problem space of \mathcal{X} by defining constraints on variables in \mathcal{X} . A constraint c has a scope $X_c \subseteq \mathcal{X}$ and a ‘relation’ which we express by a set R_c of assignments to the scope X_c . This set can be specified explicitly in form of a table where each column corresponds to a variable in X_c , each row corresponds to an assignment in R_c , and the value v from a value assignment $(x = v) \in \sigma$ is put in the cell for column x and row σ . The relation R_c can also be specified by a logical formula that involves the variables from X_c and the operations from a given mathematical structure over the variable domains. A *constraint satisfaction problem* CSP for \mathcal{X} is given by a finite set of constraints \mathcal{C} the scopes of which are all subsets of \mathcal{X} . A CSP is finite if all its domains and relations are finite. A constraint c is satisfied by an assignment σ to \mathcal{X} iff $\sigma[X_c]$ is an element of R_c . An assignment σ is a *solution* of \mathcal{C} iff it satisfies all constraints of \mathcal{C} . If a CSP has no solution then it is called inconsistent. It is often convenient to replace a CSP \mathcal{C} by the conjunction $\bigwedge_{c \in \mathcal{C}} c$ of its constraints. Hence all definitions and propositions referring to constraints C_1, C_2 can also be applied to CSPs.

We now define preference orderings on a problem space. In decision theory, preferences can be modelled in form of a complete preorder \succsim that can be decomposed into a strict part \succ and indifference \sim . In this paper, we are only interested in strict preferences and we suppose that these preferences can be specified in an incomplete way, which allows us to refine them. The absence of a strict preference between two alternatives thus can either signify indifference or incompleteness. Hence, we specify preferences in form of a strict partial order \succ on the assignments to \mathcal{X} . We write $\sigma_1 \succeq \sigma_2$ as a short-hand for $\sigma_1 \succ \sigma_2$ or $\sigma_1 = \sigma_2$. We also consider the inverse order \prec . We have $\sigma_1 \prec \sigma_2$ iff $\sigma_2 \succ \sigma_1$. We say a solution σ of C is a \succ -best (\succ -worst or \prec -best) solution of C iff there is no other solution σ^* of C s.t. $\sigma^* \succ \sigma$ ($\sigma^* \prec \sigma$).

Although a strict partial order \succ on the assignments to \mathcal{X} has an exponential size in general, there are interesting cases allowing a polynomial representation, in particular if a subset X of variables is preferentially independent from its complement. In this case, $A \cup C \succ B \cup C$ is equivalent to $A \cup D \succ B \cup D$ for all assignments A, B to X and all assignments C, D to $\mathcal{X} - X$. The projection $\succ_{[X]}$

of the partial order \succ on the variables X is well-defined in this case. We simply define $A \succ_{[X]} B$ iff there exists C s.t. $A \cup C \succ B \cup C$.

Pareto-dominance naturally meets this independence condition. Given two strict partial orders \succ_1, \succ_2 on two disjoint sets of variables X_1, X_2 , we say that an assignment σ_1 to $X_1 \cup X_2$ dominates an assignment σ_2 to $X_1 \cup X_2$ in the Pareto-sense iff 1. σ_1 is different from σ_2 and 2. σ_1 is better than or equal to σ_2 on X_1 (i.e. $\sigma_1[X_1] \succeq_1 \sigma_2[X_1]$) and σ_1 is better than or equal to σ_2 on X_2 (i.e. $\sigma_1[X_2] \succeq_2 \sigma_2[X_2]$). We can apply this combination in an inductive manner to n strict partial orders, namely one for each variable in \mathcal{X} .

We also consider the case where the strict partial order is the strict part of a complete preorder. It is called a *ranked order* in this case, which is a strict partial order satisfying the following property for all assignments $\sigma_1, \sigma_2, \sigma_3$: if $\sigma_1 \succ \sigma_2$ then either $\sigma_3 \succ \sigma_2$ or $\sigma_1 \succ \sigma_3$. Ranked orders can be represented by utility functions u that map assignments to a numerical value s.t. $u(\sigma_1) > u(\sigma_2)$ iff $\sigma_1 \succ \sigma_2$.

3 Preference-based Separation

Given a set of constraints \mathcal{C} , we either want to prove that \mathcal{C} has no solution or find a solution of \mathcal{C} . Systematic search methods solve this problem by exploring alternative ways to tighten it, e.g. by choosing a variable x and by adding the constraint $x = v$ for different values v to \mathcal{C} . An alternative way is to relax certain constraints by using some preference order \succ . For example, if we want to book a hotel for each guest, but the cheapest hotel exceeds the budget of the richest guest, then there is no solution to the problem. Here, we relax some of the constraints in a way such that we consider only the \succ -best solutions of those constraints and we relax other constraints such that we consider only the \succ -worst solutions of those constraints. To formalize this, we use a preference order \succ to define a relaxation of a constraint C , i.e. a constraint C' such that $R_C \subseteq R_{C'}$:

Definition 1 Let \succ be a strict partial order and C a constraint. We define the \succ -relaxation C^\succ as the set of all assignments that are equal or worse (w.r.t. the order \succ) than a solution of C .

We can uniquely characterize a \succ -relaxation as follows:

Proposition 1 The \succ -relaxation of C is the greatest relaxation of C that has the same \succ -best solutions as C .

We now consider two constraints C_1 and C_2 and want to determine whether they have a common solution. Given an order \succ , we perform this test for the \succ -relaxation of C_1 and the \prec -relaxation of C_2 . Hence, the same order is used to determine a relaxation of C_1 and of C_2 . Figure 1 illustrates these relaxations for a Pareto-order obtained by combining the increasing order on x with the increasing order on y . We show the Pareto-frontier which is spanned by the \succ -best solutions of C_1 and the Pareto-frontier which is spanned by the \prec -best solutions of C_2 . Both relaxations have a common solution iff the Pareto-frontiers intersect:

Proposition 2 (Separation principle) Let \succ be a strict partial order and C_1, C_2 two constraints. $C_1^\succ \wedge C_2^\prec$ is consistent if and only if there is a \succ -best solution σ_1 of C_1^\succ and a \succ -worst solution σ_2 of C_2^\prec such that $\sigma_1 \succeq \sigma_2$.

Proof: (\Rightarrow) If σ is a solution of $C_1^\succ \wedge C_2^\prec$ then it is \succ -worse than or equal to a \succ -best solution σ_1 of C_1^\succ and \succ -better than or equal to a \succ -worst solution σ_2 of C_2^\prec . Hence, we get $\sigma_1 \succeq \sigma \succeq \sigma_2$. (\Leftarrow) Since σ_1 is a best solution of the \succ -relaxation C_1^\succ and σ_2 is \succ -worse than σ_1 , the solution σ_2 of C_2^\prec satisfies C_1^\succ as well, meaning that the conjunction of both constraints is consistent. \square

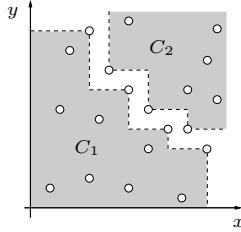


Figure 1. Pareto-dominance as a separating order.

As a corollary, we can state that if no \succ -best solution σ_1 of C_1 and no \succ -worst solution σ_2 of C_2 satisfy $\sigma_1 \succeq \sigma_2$, then $C_1 \wedge C_2$ is inconsistent. This gives a sufficient condition for inconsistency, but not a necessary one. If an order \succ satisfies this sufficient condition, then we call it a *separating order* for C_1 and C_2 . For ranked orders we can limit the test for separating orders to a single solution:

Proposition 3 *Let \succ be a ranked order. If a \succ -best solution of C_1 is strictly worse than a \succ -worst solution of C_2 then all \succ -best solutions of C_1 are strictly worse than all \succ -worst solutions of C_2 .*

As an example for a ranked separating order, consider two constraints $x = 2 \cdot y + 7$ and $x = 2 \cdot y + 4$ on integer variables x, y . Let $\sigma_1 \succ \sigma_2$ iff σ_1 assigns an even value to x and σ_2 assigns an odd value to x .

It is easy to show that a separating order for C_1 and C_2 exists whenever $C_1 \wedge C_2$ is inconsistent. We can, for example, label the solutions of C_1 by 1 and the solutions of C_2 by -1 and use this label as ranking function. Those labels provide a proof why no solution exists. For establishing this proof, we still need to compute each solution of C_1 and each solution of C_2 . Hence, we do not gain much for inconsistency proving if the separating order is defined in terms of both C_1 and C_2 and depends on the combination of both constraints. In the next section, we explore cases where a separating order can be defined in terms of only one of the constraints.

4 The Best-Fail Principle

In many cases, the solution space of a constraint satisfies some global properties. If an assignment $\{x = v, y = w\}$ satisfies the constraint $x \geq y$ then we know that all other assignments $\{x = v', y = w'\}$ with $v' \geq v$ and $w \geq w'$ satisfy the constraint as well. Those dependencies can be characterized in form of a preference order on the problem space. We prefer assignments that satisfy the constraint to assignments that violate it. In other words, if an assignment satisfies a constraint then all better assignments need to satisfy it.

Definition 2 *A strict partial order \succ is a preference order for a constraint C iff all assignments σ_1, σ_2 satisfy the following property: if $\sigma_1 \succ \sigma_2$ and σ_2 satisfies C then σ_1 satisfies C .*

Hence, the truth function of a constraint is anti-monotonic w.r.t. its preference order. Preference orders capture dominance phenomena and classical styles of relaxations such as Lagrangian relaxation.

The constraint $e(x) \leq b$ has the preference order \succ_e that prefers smaller values of the expression $e(x)$. Hence, $\sigma_1 \succ_e \sigma_2$ holds iff $(x = v_1) \in \sigma_1, (x = v_2) \in \sigma_2$, and $e(v_1) < e(v_2)$. This case covers linear and non-linear inequalities. Multiple inequalities of the form $\bigwedge_i e_i(x) \leq b_i$ have the Pareto combination of $\succ_{e_1}, \dots, \succ_{e_n}$ as a preference order, but the order \succ_e obtained from an additive sum $e(x) := \sum_i \lambda_i \cdot e_i(x)$ is not a preference order for this conjunction.

Conditional expressions such as $\text{if } c(y) \text{ then } e_1(x) \text{ else } e_2(x)$ can be handled by context-dependent preferences as expressible in CP-nets. We can combine preference orders for boolean expressions:

Proposition 4 *If \succ is a preference order for constraint C then \prec is a preference order for $\neg C$. If \succ_1 is a preference order for constraint C_1 and \succ_2 is a preference order for constraint C_2 , then $\succ_1 \cap \succ_2$ is a preference order for $C_1 \wedge C_2$ and for $C_1 \vee C_2$.*

We also obtain a second characterization of \succ -relaxations.

Proposition 5 *A constraint C is equal to its \prec -relaxation C^\prec iff \succ is a preference order of C .*

Hence, preference orders can directly be used as separating orders. Interestingly, a preference order \succ of C_2 is a separating order for C_1 and C_2 and this is independent of the properties of C_1 . Furthermore, we can simplify the separation principle for preference orders. It is sufficient to check whether the best solutions of C_1 satisfy C_2 .

Proposition 6 (Best-fail principle) *Let \succ be a preference order for C_2 . $C_1 \wedge C_2$ is consistent if and only if there is a \succ -best solution σ^* of C_1 that satisfies C_2 .*

Proof: (\Rightarrow) Let σ be a solution of $C_1 \wedge C_2$. There exists a \succ -best solution σ^* of C_1 such that $\sigma^* \succeq \sigma$. We show that σ^* satisfies C_2 by distinguishing two cases: 1. If σ^* is equal to σ then σ^* satisfies C_2 since σ does so. 2. If σ^* is different from σ then $\sigma^* \succ \sigma$. Since \succ is a preference order for C_2 and σ satisfies C_2 , σ^* must satisfy C_2 . (\Leftarrow) since σ^* satisfies both C_1 and C_2 the conjunction $C_1 \wedge C_2$ is consistent. \square

For ranked orders, it is sufficient to check a single solution.

Proposition 7 *Let \succ be a ranked preference order for C_2 . $C_1 \wedge C_2$ is inconsistent iff there is a \succ -best solution of C_1 that violates C_2 .*

Hence, to prove the inconsistency of $C_1 \wedge C_2$ it is sufficient to show that all \succ -best solutions of C_1 violate C_2 . We thus reduce the unsatisfiability problem to an optimization problem. An inconsistency proof can be represented by the set of \succ -best solutions of C_1 (or a single best solution for ranked orders).

The best-fail principle can immediately be applied to scheduling and configuration problems that are inconsistent due to tight bound constraints. For example, job-shop scheduling problems are guaranteed to have solutions if all due date constraints are relaxed. Similarly, it is a natural assumption that product configuration problems without user requests have solutions and user requests have the form of bound constraints on global criteria in many interesting cases.

The best-fail principle also generalizes the way in which LP solvers prove the unsatisfiability of overconstrained linear programs. The solver does not solve the original problem, but a dual problem where all constraints are relaxed and the objective consists in maximizing the satisfaction of the constraints. We can generalize this dual solving by the following sequential process. Initially, we relax all of the given n constraints c_1, \dots, c_n . We will step by step add constraints to a pool C_i and maintain a solution σ_i of the pool C_i . We start with the empty pool $C_0 = \emptyset$ and an arbitrary assignment σ_0 . In the i -th step, we try to add the constraint c_i if it is consistent w.r.t. the pool C_{i-1} . To test this, we generate a preference order \succ_i for c_i and determine the \succ_i -best solutions of C_{i-1} . If they all violate c_i then we have proved that the constraints c_1, \dots, c_n are inconsistent. If one of the optimal solution satisfies c_i , then we add c_i to the pool by doing $C_i := C_{i-1} \cup \{c_i\}$ and we set σ_i to this solution.

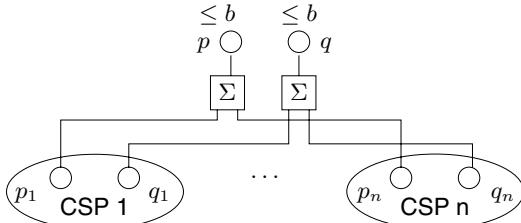


Figure 2. Configuration problem with resource constraints

When solving the optimization subproblem, we can profit from the fact that σ_{i-1} is a solution of C_{i-1} . We can use this as a starting point for the optimization. Indeed, the problem consists in finding one (or all) solutions of C_{i-1} that improve a solution σ w.r.t. the preferences \succ_i . This is a kind of generalized pivot operation and the problem can be solved efficiently for linear constraints (and other constraints having a convex shape). Hence, the best-fail principle provides an effective generalization of the ‘dual’ constraint solving iff all constraints have suitable preference ordering and the optimization subproblem can efficiently be solved for those preferences.

5 Best-Fail for Global Resource Constraints

As optimization problems are often solved by a sequence of satisfiability problems, it is not evident to ensure that the optimization problem C_1 can be solved faster than solving the original satisfiability problem $C_1 \wedge C_2$. A significant speed-up can be obtained for the optimization problem iff this problem breaks down into several independent subproblems. This is the case for an important class of configuration problems.

Many configuration problems (e.g. simple shopping baskets) involve the configuration of multiple key components. For example, my audio/video system may consist of a TV, DVD player, a tuner, and so on. Those key components are designed in a way such that we can choose them from different product catalogues and combine them without facing technical incompatibilities. Each key component may have a huge number of possible configurations and we would like to configure each of these components independent of the other ones in order to obtain an additive complexity. However, there are non-technical constraints that need to be respected by the whole system. These are the user requirements on global properties such as the total price and the total quality of the system.

We illustrate the difficulties of these configuration problems by using a fictive travel planning example. It corresponds to an extreme case which is nevertheless instructive and its pattern can occur in more complex scenarios: *Space voyager LJ will visit the planets 1 and 2 of the All-Value-Galaxy and seeks a Hotel close to a sea resort on both planets. Hotels in the All-Value-Galaxy try to distinguish themselves by different prices, qualities, beach distances and so on. Due to the huge number of hotels, we can find a hotel for nearly every value of each criterion as long as this value is within a given range. And moreover, we can find all price/beach-distance combinations within a given region in the problem space except for some exceptions. Figure 3 shows those regions for the planets 1 and 2. The forbidden combinations are crossed out. LJ has a total budget of $b := 1200$ Star dollars and wants to keep the total sum of beach distances smaller than 1200 m. According to Figure 3, planet 1 has hotels for 300 dollars and planet 2 has hotels for 200 dollars. Hence the global budget constraint can be met if we ignore the beach-distance*

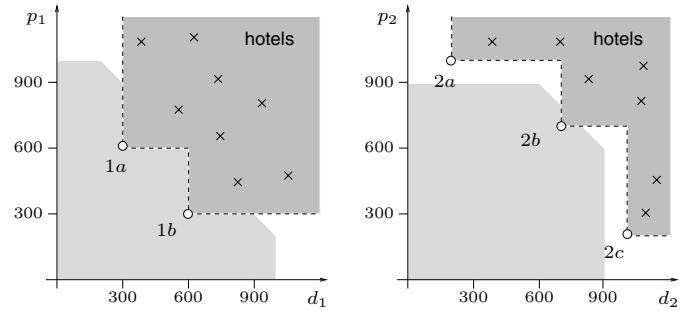


Figure 3. Price-distance constraint for planet 1 (left) and planet 2 (right).

requirement. Similarly, we can meet the beach distance requirement if we forget about the budget. But can we meet both requirements?

To answer this question, we set up a CSP. Since the only dependencies between the different hotel choices are the global resource constraints we obtain a structured CSP as depicted in Figure 2. It consists of n sub-CSPs, namely one for each key component. In the general case, the sub-CSP can involve multiple variables and constraints. In our simple example, the i -th sub-CSP consists of a single constraint representing the hotel catalogue of the i -th planet. It has variables such as product identifier, price, quality, and so on. The prices of the different components are summed up to compute the total price which is submitted to user requirements (e.g. an upper bound on the total price). We obtain a similar resource constraint for the total distance.

A naive method to solve the problem will determine the set of all configurations for each component and then tries to combine them in a way to meet the global resource constraints. In our example, the configurations for a single component are already provided by the t catalog entries. The number t is close to b^2 in the considered example. We then obtain a multiplicative complexity of $O(t^n)$ in order to solve the entire problem which includes the global resource constraints. Our purpose is to achieve a complexity of $O(n \cdot t)$ which is additive for the subproblems.

If there is a single global resource constraint, such an additive approach is easy to figure out. Suppose that there is only one global requirement $\sum_j x_j \leq b$, say on the price. Then the CSP is inconsistent if and only if the sum of the cheapest solutions of each sub-CSP exceeds the given maximum price. Hence, it is sufficient to determine the cheapest entry of each product catalogue, which is linear in the number t of catalogue entries (or even faster if the product catalogue is available in form of a data-base with suitable indexing structure). In this case, we have mapped the unsatisfiability problem into n optimization subproblems by relaxing the maximum price constraint. Each optimization subproblem can be solved in $O(t)$ which leads to the desired additive complexity.

Multiple requirements of the form $y_i \leq b_i$ where $y_i = \sum_j x_{i,j}$ lead to multiple criteria y_i each of which need to be minimized. There are different ways to combine these criteria to a preference order on assignments. We are not satisfied by an arbitrary aggregation of the individual preferences, but want to establish a preference order for the sum-constraint as required by definition 2.

A popular method is the Lagrangian relaxation, which seeks a positive weight λ_i for each criterion and uses the additive utility function $\sum_i \lambda_i \cdot y_i$ to define the global order. However, this leads to a large relaxation which may indeed intersect with the non-relaxed constraints although no solution exists. We can also look at the convex hull of

the relaxed constraints, but it is not easy to define the corresponding order. Indeed, we first need to compute the convex hull and then define the order. Again, this relaxation may be too large as demonstrated by the example. First, we determine the convex hull of the hotels on planet 1. We are then interested in the region of all points $(b - v, b - w)$ which is left over if any hotel (v, w) from this convex hull is chosen. It is clear that a hotel from planet 2 needs to fall into this second region to guarantee the existence of a solution. The right part of Figure 3 shows this region in light Grey. It contains the hotel $2b$, meaning that there might be a solution. Hence, the convex hull is not sufficient to prove the unsatisfiability of our example.

The Pareto-order does not have these problems since it is a preference order of a conjunction of inequalities of the form $y_i \leq b_i$. Hence, it leads to a relaxation that is equal to the solution space of this conjunction. As for the classical relaxations based on a single criterion, there is no solution if the solution space of the relaxed constraints does not intersect with the non-relaxed constraints. In contrast to the Lagrangian relaxation, the other direction holds as well: *if there is no solution then there is no intersection*.

Pareto-dominance thus allows us to apply the best-fail principle if we want to relax multiple resource constraints. In the example, we relax both the price and distance requirements simultaneously. We then need to determine all combinations of component configurations which are Pareto-optimal w.r.t. the minimization of the four variables p_1, p_2, d_1, d_2 . Since the global constraints are all relaxed, the non-relaxed constraints split into n independent subproblems based on the following result:

Proposition 8 *Let \mathcal{X} be a set of variables. Let C_1, \dots, C_n be constraints on variables from \mathcal{X} such that C_i and C_j have disjoint variables for $i \neq j$. Suppose that \succ is a strict partial order such that $X(C_i)$ is preferentially independent from $\mathcal{X} - X(C_i)$ for all i . Then σ is a \succ -preferred solution of $\bigwedge_i C_i$ iff $\sigma[C_i]$ is a $\succ_{[X(C_j)]}$ -preferred solution of C_i for all $i = 1, \dots, n$.*

Hence, the decomposition is possible since the n subproblems do not have any variable in common and since the Pareto-dominance ensures that the price and distance variables of each subproblem are preferentially independent from the price and distance variables of the other subproblems.

The first subproblem consists in finding all hotels of planet 1 with minimal price and with minimal beach distance. All Pareto-optimal solutions with respect to both criteria need to be found. A similar task need to be solved for planet 2. For planet 1, we find two Pareto-optimal solutions, namely $(600, 300)$ and $(300, 600)$. For planet 2, we find three Pareto-optimal solutions, namely $(200, 1000)$, $(700, 700)$, and $(1000, 200)$. Any combination of those Pareto-optimal solutions leads to a Pareto-optimal solution for the conjunction of the sub-CSP1 and sub-CSP2 and the other direction works as well. Hence, the best solutions of the non-relaxed problem are given by:

$$\left(\begin{array}{l} (p_1 = 600 \wedge d_1 = 300) \vee \\ (p_1 = 300 \wedge d_1 = 600) \end{array} \right) \wedge \left(\begin{array}{l} (p_2 = 200 \wedge d_2 = 1000) \vee \\ (p_2 = 700 \wedge d_2 = 700) \vee \\ (p_2 = 1000 \wedge d_2 = 200) \end{array} \right)$$

It is easy to see that this is inconsistent w.r.t. $p_1 + p_2 \leq 1000$ and $d_1 + d_2 \leq 1000$. Hence, if the number of Pareto-optimal solutions of each subproblem is small, the problem breaks down. We still need to test all combinations of Pareto-optimal solutions (six in our example), but we can, for example, exploit arc consistency to filter the Pareto-optimal solutions of the subproblems. The following theorem applies the best-fail principle to decomposable problems and thus summarizes the approach of this section:

Theorem 1 *Let \mathcal{X} be a set of variables. Let C, C_1, \dots, C_n be constraints on variables from \mathcal{X} such that C_i and C_j have disjoint variables for $i \neq j$. Suppose that \succ is a preference order for C for which $X(C_i)$ is preferentially independent from $\mathcal{X} - X(C_i)$ for all i . The conjunction $C \wedge \bigwedge_i C_i$ is inconsistent iff there are no $\succ_{[X(C_j)]}$ -preferred solutions σ_i of C_i s.t. $\bigcup_i \sigma_i$ satisfies C .*

We also explain the cost for finding the Pareto-optimal solutions. In our example, we have two criteria and can thus use the algorithm from [6] (generalized to multiple criteria in [3]). It can be applied as follows: first we seek the best price p_1 starting from 1. If there is no support for the current price we increase it. Once we found the best price, we seek the best distance among the supports for this price. The result of this gives the first Pareto-optimal solution. We then check whether an increase of the price can reduce the distance. We repeat this procedure until no further increase of the price brings a reduction of the distance. We thus need to scan once over all possible values of the price and over all possible supports of each price. Hence, the complexity for computing the Pareto-frontier of one constraint is bounded by $O(b + t)$.

6 Conclusion

We have addressed an important limitation of classical relaxation techniques from Operations Research. Although those techniques are very effective in proving the unsatisfiability of overconstrained problems by relaxing some of the constraints and by replacing them by an optimization objective, they impose unnecessary restrictions on the form of the relaxation. In this paper, we profit from recent work on preferences in AI and in databases and develop a relaxation technique based on general partial orders. Based on this, we are able to use Pareto-frontiers to test the satisfiability of global resource constraints for product configuration problems. Proof complexity is rather reduced compared to pure AC-based methods and more comprehensive explanations for unsatisfiability are obtained. There is room to improve the approach and to reduce the Pareto-frontier in size, for example by further relaxing the constraints or by using indifference among values to tighten the preference orders of the constraints [5]. Furthermore, our preference-based relaxation framework also promises to produce relaxations for other forms of constraints such as conditional expressions.

Acknowledgments

I would like to thank the anonymous reviewers as well as Hassan Aït-Kaci and Olivier Lhomme for very helpful comments.

REFERENCES

- [1] Christian Bessière, Anaïs Fabre, and Ulrich Junker, ‘Propagate the right thing: how preferences can speed-up constraint solving’, in *IJCAI’03*, pp. 191–196, Acapulco, (2003).
- [2] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, ‘Preference-based constrained optimization with CP-nets’, *Computational Intelligence*, **20**, 137–157, (2004).
- [3] Ulrich Junker, ‘Preference-based search and multi-criteria optimization’, *Annals of Operations Research*, **130**, 75–115, (2004).
- [4] Werner Kießling, ‘Foundations of preferences in database systems’, in *28th International Conference on Very Large Data Bases (VLDB 2002)*, pp. 311–322, (2002).
- [5] Werner Kießling, ‘Preference queries with SV-semantics’, in *COMAD*, pp. 15–26, (2005).
- [6] Luc N. Van Wassenhove and Ludo F. Gelders, ‘Solving a bicriterion scheduling problem’, *European Journal of Operational Research*, **4**(1), 42–48, (1980).

Return of the JTMS: Preferences Orchestrate Conflict Learning and Solution Synthesis

Ulrich Junker and Olivier Lhomme¹

Abstract. We use a lexicographical preference order on the problem space to combine solution synthesis with conflict learning. Given two preferred solutions of two subproblems, we can either combine them to a solution of the whole problem or learn a ‘fat’ conflict which cuts off a whole subtree. The approach makes conflict learning more pervasive for Constraint Programming as it well exploits efficient support finding and compact representations of Craig interpolants.

1 Introduction

A justification-based truth maintenance system (JTMS) [6] allows a problem solver to record logical dependencies between deductions. A JTMS supports the retraction of constraints and provides explanations for the recorded deductions. The JTMS also introduced a technique which is now called conflict learning [10, 20]. Conflicts permit a search procedure to backtrack to the causes of a failure, while keeping choices in independent subproblems unchanged. As a consequence, we obtain an additive complexity instead of a multiplicative complexity for independent subproblems [9]. Although independent subproblems are frequent in configuration and design problems, they are rare in popular applications of Constraint Programming (CP) such as scheduling and resource allocation. Hence, conflict learning was seldom used in CP, although recent efforts are promising [14].

However, conflict learning proved to be essential for the recent success of SAT solvers [10, 20]. It can be observed that conflict learning reduces the complexity of a problem even if subproblems are not independent, but have some variables in common. Suppose a problem solver first determines a solution S_1 of subproblem $A(x, y)$ and then solves the subproblem $B(y, z)$ under the restrictions that S_1 imposes on the common variables y . If this restricted version of B has no solution, the problem solver backtracks and seeks a new solution S_2 of the subproblem A . It then solves subproblem B under the restrictions S_2 imposes on y . Hence, the subproblem B is solved several times and this for different solutions S_i of the subproblem A . Although these solutions are all different, their projection on the common variables y can be the same, meaning that the same restricted version of the subproblem B is solved again and again. This thrashing behaviour will in particular be very cumbersome if the whole problem has no solution and the whole solution space of A needs to be traversed. Conflict learning permits the problem solver to avoid this thrashing behaviour. If a subproblem B fails for a restriction C on the common variables y , the conflict $\neg C$ is incorporated into the problem A and avoids that solutions with restriction C on variables y are generated again. If the whole problem is inconsistent, then the set of learned conflicts is a Craig interpolant [4, 16]. Conflict learning thus permits a factorization of the problem solving. We can also

show that conflict learning behaves like a lazy version of bucket elimination [5] if the buckets are chosen as subproblems. Hence, conflict learning is exponential in the induced width [5, 8] in the worst-case.

JTMS thus well supports factorization of problems provided it uses a static preference ordering between variables as proposed in [17] when choosing the culprits to be retracted. Moreover, JTMS can be used to maintain a lexicographically preferred solution. As described in [6], non-monotonic justifications can be used to encode preferences between the different values of a variable, which leads to an ordered disjunction [3]. This mechanism always activates the best assignment for a variable that is not defeated by a recorded conflict. If this best assignment occurs in a new conflict, JTMS can retract this assignment and activate the next one. This procedure will finally result in a lexicographically preferred solution, which will be updated if new constraints are added (e.g. new conflicts learned from interactions with other problems). However, if variable domains are large, then this procedure may try out many value assignments. An activated value assignment $y = w$ will only be abandoned if it occurs in a conflict. When seeking a support for an assignment $x = v$ on the constraint $C(x, y)$, d such assignments $y = w_i$ may be tried out, meaning that d conflicts have to be learned.

Modern algorithms that maintain arc consistency (MAC) [19] directly seek supports (i.e. solutions of a single constraint) without needing to learn so many conflicts. Moreover, the preferences between values can be used to activate preferred supports [2]. Provided with an assignment $x = v$, the constraint $C(xy)$ can thus determine the preferred support (v, w) for $x = v$ and activate the assignment $y = w$ without trying out all better value for y . An even further idea is to synthesize the preferred supports from different constraints into a lexicographically preferred solution. PrefAC [2] has this behaviour for tree-like CSPs if parent variables are preferred to child variables. However, the activation of preferred supports gets blocked for cyclic constraint networks when different preferred supports activate different values for the same variable.

It is possible to learn a conflict in those situations. In this paper, we show how to combine solution synthesis and conflict learning. Consider again the two subproblems $A(xy)$ and $B(yz)$. We first determine a preferred solution S_1 for A . We then determine a preferred solution S_2 for B under the restriction that it must not be better than S_1 on the common variables y . If both solutions assign the same values to the common variables y , then we can synthesize them to a solution of the whole problem. Otherwise, we learn a ‘fat’ conflict that cuts off a whole subtree in the search space of y . All assignments between S_1 (projected to y) and S_2 (projected to y) are infeasible. Hence, less solutions of A need to be generated in turn and conflict learning becomes effective for CSPs even if domains are large. By introducing fat conflicts, we enable a return of the JTMS to CP.

¹ ILOG S.A., France, email: {ujunker, olhomme}@ilog.fr

The paper is organized as follows: Firstly, we introduce solution synthesis and conflict learning based on a lexicographical order and show how they can be used to decompose a problem into two subproblems. Secondly, we apply this decomposition recursively until we obtain buckets, i.e. constraints having a variable in common. We show how to solve the buckets by reducing lexicographical bounds.

2 Preference-based Interpolation

2.1 Constraints and Preferences

Throughout this paper, we consider combinatorial problems that are modelled by a set of constraints C formulated over a set of variables X having the domain \mathcal{D} . A variable x from X can be interpreted by any value v from the domain \mathcal{D} and we describe such a variable-value assignment by the constraint $x = v$. An *assignment* to X is a set that contains exactly one value assignment for each variable in X . The set of all assignments to X is called the *problem space* of X , and we denote it by $\mathcal{S}(X)$. We project an assignment σ to a set Y by defining $\sigma[Y] := \{(x = v) \in \sigma \mid x \in Y\}$. If the assignment σ contains $x = v$, we denote the value v of x by $\sigma(x)$. Furthermore, we define $\sigma(x_1, \dots, x_n)$ as the vector $(\sigma(x_1), \dots, \sigma(x_n))$.

A constraint c is specified by an n -ary set of variables X_c , called scope, and an n -ary ‘relation’ R_c , which is described by a subset of $\mathcal{S}(X_c)$. The relation can be specified by a predicate, a numerical or logical expression, a set of allowed tuples, or a set of forbidden tuples. An assignment σ to X is a *solution* of the constraint c iff $\sigma[X_c] \in R_c$ holds.

The set of variables of a set of constraints C is denoted by $X_C := \bigcup_{c \in C} X_c$. An assignment σ (to X) is a *solution* of the constraints C iff it is a solution of all constraints $c \in C$. It is often convenient to replace a constraint set C by the conjunction $\bigwedge_{c \in C} c$. Similarly, we can replace a conjunction $\bigwedge_{i=1}^k c_i$ be the set of its conjuncts $\{c_1, \dots, c_k\}$. All statements about the constraints C_1, C_2 can thus be applied to CSPs and vice versa. Two constraints C_1 and C_2 with same scope are equivalent iff they have the same solutions.

Throughout this paper, we assume that the user can express preferences between the possible values of a variable x_i and we consider some linearization $>_i$ of those preferences as indicated in [13]. Furthermore, we suppose that the user can express an importance ordering between the variables and we consider a linearization $>$ of this importance ordering (cf. [13]). The ordering $>$ can be used to sort a set of variables. Let $X^>$ denote the sorted vector $(x_{\pi_1}, \dots, x_{\pi_n})$ where X is equal to $\{x_1, \dots, x_n\}$ and π is the permutation of $1, \dots, n$ that satisfies $x_{\pi_1} > \dots > x_{\pi_n}$. We define the lexicographical ordering $>_{lex}$ between two vectors of values $v := (v_{\pi_1}, \dots, v_{\pi_n})$ and $w := (w_{\pi_1}, \dots, w_{\pi_n})$: $v >_{lex} w$ holds iff there exists an i among $1, \dots, n$ such that $v_{\pi_j} = w_{\pi_j}$ for $j = 1, \dots, i-1$ and $v_{\pi_i} >_{\pi_i} w_{\pi_i}$. A solution σ of C is called *preferred solution* of C iff there is no other solution σ^* s.t. $\sigma^*(X^>) >_{lex} \sigma(X^>)$. As in [13], we use an optimization operator $\text{Lex}(X^>)$ to describe the set of preferred solutions. The operator maps the constraint C to a new constraint $\text{Lex}(X^>)(C)$ that is satisfied by exactly the preferred solution of C . This notation allows us to reason about optimization results inside the constraint language.

The lexicographical ordering can be used to formulate a constraint in the constraint language. Let x be a sorted vector of variables and v be a sorted vector of values. The constraint $x >_{lex} v$ is equivalent to

$$\bigvee_{i=1}^n \left(\bigwedge_{j=1}^{i-1} x_{\pi_j} = v_{\pi_j} \wedge x_{\pi_i} >_{\pi_i} v_{\pi_i} \right) \quad (1)$$

We obtain similar constraints for \geq_{lex} , $<_{lex}$, and \leq_{lex} . The lexicographical constraint allows us to express a property which is fundamental for this paper. If σ^* is a solution of $\text{Lex}(X^>)(C)$ then C implies $X^> \leq_{lex} \sigma^*(X^>)$.

2.2 Solution Synthesis and Fat Conflicts

We now consider two sets of constraints C_1 and C_2 . Each set represents a subproblem. We will show how the whole problem $C_1 \cup C_2$ can be solved by solving the two subproblems independently. The only information that we exchange between the two subproblems are supplementary constraints that are formulated on the common variables of C_1 and C_2 , i.e. $X_{C_1} \cap X_{C_2}$. If the whole problem $C_1 \cup C_2$ is inconsistent, then it is well-known that there is a constraint C' that contains only the common variables, that is implied by C_2 , and that is inconsistent w.r.t. C_1 . The constraint C' is called a *Craig interpolant* [4, 16]. In this section, we will show that we can either synthesize two solutions of the subproblems into a solution of the whole problem or compute a Craig interpolant by generating a sequence of conflicts. We use the lexicographical preference ordering to determine which solutions and conflicts may be generated in each step.

We can only synthesize two solutions of the subproblems if the importance ordering $>$ on the variables ensures that the variables added by C_2 are less important than the variables in C_1 . We first lift the ordering on variables to sets of variables: $X_1 \succ X_2$ iff $x_i > x_j$ for all $x_i \in X_1, x_j \in X_2$. We then extend it to constraints:

Definition 1 A constraint C_1 is more important than a constraint C_2 in the ordering $>$, written $C_1 \succ C_2$, iff $X_{C_1} \succ X_{C_2} - X_{C_1}$.

An example is given be the two constraints in table 1. The domain is $\mathcal{D} := \{1, 2, 3\}$. Greater values are preferred for all variables and the variables are ordered as follows: $x_1 > x_2 > x_3 > x_4$.

From now on, we suppose that C_1 is more important than C_2 . In section 3.1, we will show that we can always find subproblems that meet this condition. In the sequel, we write X_1 for the ordered tuple of variables $X_{C_1}^>$ of C_1 , X_2 for the ordered tuple of variables $X_{C_2}^>$ of C_2 , Δ for the ordered tuple of variables $(X_{C_1} \cap X_{C_2})^>$ occurring in both subproblems, and Ω for the ordered tuple of variables $(X_{C_1} \cup X_{C_2})^>$ occurring in the whole problem.

We start the whole solving process by seeking a preferred solution for the first subproblem C_1 . If this problem is inconsistent, then $C_1 \cup C_2$ is inconsistent as well. Hence, the failure of the first subproblem leads to a global failure and we can stop in this case.

If the first subproblem has the preferred solution σ_1 , then we project it on the common variables by taking $\sigma_1(\Delta)$. If we want to find a solution σ_2 of the second subproblem that is compatible with σ_1 , then this other solution cannot be lexicographically better than σ_1 on the common variables Δ . Indeed, if $\sigma_2(\Delta) >_{lex} \sigma_1(\Delta)$ holds, then the two solutions cannot be synthesized. We therefore temporarily add a lexicographical constraint $\Delta \leq_{lex} \sigma_1(\Delta)$ to C_2 when determining a preferred solution of the second subproblem. Please note that this constraint involves only variables that are shared by both subproblems. If the enforced version of the second subproblem has no solution, then the added constraint cannot be true and we can add its negation to the first subproblem. We thus obtain a first part of an interpolant:

Proposition 1 Let C_1, C_2 be two constraint sets and X_1, Δ as defined above. Let σ_1 be a solution of $\text{Lex}(X_1)(C_1)$. If $C_2 \cup \{\Delta \leq_{lex} \sigma_1(\Delta)\}$ is inconsistent, then

$$C_1^* := C_1 \cup \{\Delta >_{lex} \sigma_1(\Delta)\}$$

Table 1. Two constraints $C_1(x_1, x_2, x_3)$ and $C_2(x_2, x_3, x_4)$

x_1	x_2	x_3		x_2	x_3	x_4	
3	2	2	(1. σ_1)	3	3	2	
3	1	3		3	1	3	(4. σ'_2)
1	3	2	(3. σ'_1)	2	3	3	
1	2	2		2	3	1	
1	2	1		1	2	3	(2. σ_2)
1	1	3		1	1	2	

is a constraint that is not satisfied by σ_1 and $C_1^* \cup C_2$ is equivalent to $C_1 \cup C_2$. Furthermore, $C_1^* \succ C_2$ if $C_1 \succ C_2$.

If the enforced version of the second subproblem has the preferred solution σ_2 , then we need to check whether σ_1 and σ_2 coincide on the common variables Δ . If they do, then we can synthesize the two solutions. Moreover, we also know that the result is a preferred solution as C_2 only adds less important variables. This property is important if we use the whole procedure recursively. However, if the solutions do not coincide on the common variables, then we can remove all assignments to Δ which are between $\sigma_1(\Delta)$ and $\sigma_2(\Delta)$ with respect to the lexicographical ordering. Whereas standard conflict learning only removes a branch in a subproblem, we remove a whole subtree. Hence, we learn ‘fat’ conflicts:

Proposition 2 Let C_1, C_2 be two constraint sets and X_1, X_2, Δ, Ω as defined above. Let σ_1 be a solution of $\text{Lex}(X_1)(C_1)$ and σ_2 be a solution of $\text{Lex}(X_2)(C_2 \cup \{\Delta \leq_{\text{lex}} \sigma_1(\Delta)\})$.

1. If $\sigma_1(\Delta) = \sigma_2(\Delta)$ and $C_1 \succ C_2$ then $\sigma_1 \cup \sigma_2$ is a solution of $\text{Lex}(\Omega)(C_1 \cup C_2)$.
2. If $\sigma_1(\Delta) \neq \sigma_2(\Delta)$ then

$$C_1^* := C_1 \cup \{\Delta \leq_{\text{lex}} \sigma_2(\Delta) \vee \Delta >_{\text{lex}} \sigma_1(\Delta)\}$$

is a constraint that is not satisfied by σ_1 and $C_1^* \cup C_2$ is equivalent to $C_1 \cup C_2$. Furthermore, $C_1^* \succ C_2$ if $C_1 \succ C_2$.

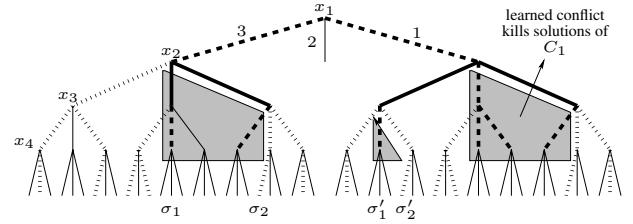
This proposition summarizes the approach of this paper. The first case corresponds to solution synthesis. The second case corresponds to conflict learning. Both are coordinated by the global preference ordering.

Compared to MAC [19], the constraints do not communicate conjunctions of domain constraints (such as $x_i \neq v_i$) to each other, but lexicographical constraints between a subset of variables and a vector of values. The result of this communication is either solution synthesis or a ‘fat’ conflict. As an effect, the number of iterations for eliminating inconsistent values can be reduced. In the example of table 1, the inconsistency of the problem is proved in the third iteration.

2.3 Lex-based Interpolation

We now develop an interpolation algorithm based on the principles developed in the last section. It basically iterates those principles until a preferred solution of the whole problem has been found or a Craig interpolant has been added to the first subproblem, thus making it inconsistent.

The algorithm LEXINTERPOLATE is shown in figure 2. In each loop iteration, it first solves the first subproblem which consists of C_1 and of an initially empty set I containing the learned conflicts. If the first subproblem has no solution, then the whole problem is inconsistent and the algorithm stops. Otherwise, it takes the preferred

**Figure 1.** Effect of conflict learning on the search tree of C_1 .

solution σ_1 of the first subproblem and formulates a lexicographical upper bound constraint on the common variables Δ and adds it temporarily to the second subproblem. If the second subproblem produces a preferred solution σ_2 that matches σ_1 then both solutions are combined into a preferred solution of the whole problem and the algorithm stops. In all other cases, a new conflict is learned and added to the interpolant. This conflict invalidates the previous solution σ_1 of the first subproblem, meaning that this solution will be revised in the next iteration. Hence, each iteration eliminates at least one assignment in the problem space of Δ meaning that the algorithm terminates for finite domains.

Theorem 1 Suppose that $C_1 \succ C_2$. The algorithm LEXINTERPOLATE(C_1, C_2) terminates after at most $O(d^{|\Delta|})$ steps. If $C_1 \cup C_2$ is consistent, then it returns a solution of $\text{Lex}(\Omega)(C_1 \cup C_2)$. Otherwise, it returns an inconsistency and the set I of the learned conflicts is a Craig interpolant for C_2, C_1 .

The Craig interpolant in the example of table 1 is as follows:

$$\begin{aligned} & ((x_2 \leq 1) \vee (x_2 = 1 \wedge x_3 \leq 2)) \vee ((x_2 > 2) \vee (x_2 = 2 \wedge x_3 > 2)), \\ & ((x_2 \leq 3) \vee (x_2 = 3 \wedge x_3 \leq 1)) \vee ((x_2 > 3) \vee (x_2 = 3 \wedge x_3 > 2)) \end{aligned}$$

Figure 1 shows how these conflicts are learned. The upper three levels of the figure show the search tree for C_1 . The lower three levels show two copies of the search tree of C_2 which are superposed with the one of C_1 . Solutions of C_1 are displayed by dashed branches and solutions of C_2 are displayed by dotted branches (straight lines are obtained when both overlap). The shadowed areas show the conflicts. The first conflict is learned in the first copy of C_2 ’s search tree. As a consequence, it also appears in the right copy. As it has been added to C_1 , it now kills several solutions of C_1 .

There are several ways to improve the algorithm. For example, we can add a constraint $X_1 \leq_{\text{lex}} \sigma_1(X_1)$ to C_1 recording the fact that there is no better solution than σ_1 . When the first problem is resolved again in later iterations, then the search can immediately start from σ_1 and the part of the search space that is lexicographically greater than σ_1 need not be searched again. Furthermore, the first subproblem may consist of several independent parts and the addition of the conflict may concern only one of those parts. We need to keep a lexicographical upper bound constraints on each independent part in order to ensure that solving complexity is additive for the independent parts and not multiplicative [9].

Another issue are functional constraints such as $x_1 + x_2 = x_3$, in particular if this constraint belongs to the second subproblem and all variables belong to Δ . When we learn a conflict for Δ , the value of x_3 depends on those of x_1 and x_2 , meaning that we can simplify the conflict by suppressing x_3 . It is therefore worth to keep track of functional dependencies between variables and to detect a subset of Δ by a conflict analysis similar to [6, 10, 20]. The elaboration of this algorithm is a topic of future work.

3 Solution Synthesis by Support Activation

3.1 The Activation Process

We now apply the algorithm LEXINTERPOLATE to an arbitrary constraint network C and to an arbitrary total ordering $>$ of the variables of C . The idea is to recursively decompose the constraints C into subproblems. We observe that decomposition only makes sense if the second subproblem has a variable that is not contained in the first subproblem. As a consequence, we stop decomposition if all the constraints in a subproblem share the last variable of the subproblem. In [5], this is called a *bucket*. There are different ways to decompose a problem into buckets. In this paper, we visit the variables of C in the ordering $X^> = (x_{\pi_1}, \dots, x_{\pi_n})$ that is produced by $>$. For each variable x_{π_i} , we define the two subproblems $C_1^{(i)}$ and $C_2^{(i)}$:

1. $C_1^{(i)}$ is the set of all constraints from C that contain only variables from $x_{\pi_1}, \dots, x_{\pi_{i-1}}$
2. $C_2^{(i)}$ is the set of all constraints from C that contain only variables from $x_{\pi_1}, \dots, x_{\pi_i}$ and that contain the variable x_{π_i} .

The two sets are disjoint. The first subproblem for the first variable $C_1^{(1)}$ is equal to the empty set. We consider the whole problem of a variable x_{π_i} , which is the union $C_1^{(i)} \cup C_2^{(i)}$ of the two subproblems. This whole problem is equal to $C_1^{(i+1)}$ if $i < n$. The whole problem of the last variable is equal to C . The second subproblem $C_2^{(i)}$ is the bucket for the variable x_{π_i} as all of its constraints contain x_{π_i} as their last variable. The first subproblem of a variable is more important than its second subproblem, i.e. $C_1^{(i)} \succ C_2^{(i)}$ for $i = 1, \dots, n$.

Each decomposition thus meets the requirements of algorithm LEXINTERPOLATE and can be solved by it. Hence, we apply LEXINTERPOLATE in a sequence, activating one variable after the other. When we activate a variable x_{π_i} for $i > 1$, we already have a solution for its first subproblem as it is equal to the whole problem of the previous variable. We just need to find a solution for the bucket of the variable while taking into account the lexicographical constraint $\Delta \leq_{lex} \sigma_1(\Delta)$ that is formulated on all variables of the bucket except for x_{π_i} . If the second subproblem alone (i.e. without lexicographical constraint) is inconsistent, then this inconsistency will only be detected after the first subproblem is solved. If the second subproblem is smaller in size than the first subproblem, we will check the consistency of the second subproblem before solving the first one.

The whole process has some similarities to bucket elimination (BE). BE processes the variables from the last one to the first one. It eliminates a variable x_{π_i} from its bucket by deriving a new constraint between the remaining variables of the bucket. This constraint is a Craig interpolant for $C_1^{(i)}$ and $C_2^{(i)}$ if the problem is inconsistent and thus corresponds to the set of conflicts learned by LEXINTERPOLATE. Otherwise, LEXINTERPOLATE learns only a part of those constraints and thus behaves like a lazy form of bucket elimination.

3.2 Heavy Supports

We now discuss how to determine a solution for the bucket $B := C_2^{(i)}$ of variable $y := x_{\pi_i}$. When searching this solution, we have already determined a preferred solution σ_1 for the problem $C_1^{(i)}$. This solution assigns a value to each variable of the bucket except for y . Let Δ be the ordered vector of all those variables, i.e. $\Delta := (X_B - \{y\})^>$. As explained in section 2.2, we add a lexicographical upper bound constraint $\Delta \leq_{lex} \sigma_1(\Delta)$ when seeking a preferred solution of the bucket. This constraint can easily be extended to an equivalent

Algorithm LEXINTERPOLATE(C_1, C_2)

```

1.    $I := \emptyset;$ 
2.   while true do
3.      $\sigma_1 := Lex(X_1)(C_1 \cup I);$ 
4.     if  $\sigma_1$  is an inconsistency then return  $\perp$ ;
5.      $\sigma_2 := Lex(X_2)(C_2 \cup \{\Delta \leq_{lex} \sigma_1(\Delta)\});$ 
6.     if  $\sigma_2$  is an inconsistency then
7.        $I := I \cup \{\Delta >_{lex} \sigma_1(\Delta)\};$ 
8.     else if  $\sigma_1(\Delta) = \sigma_2(\Delta)$  then
9.       return  $\sigma_1 \cup \sigma_2;$ 
10.    else  $I := I \cup \{\Delta \leq_{lex} \sigma_2(\Delta) \vee \Delta >_{lex} \sigma_1(\Delta)\};$ 

```

Figure 2. Algorithm for lexicographic interpolation.

constraint that covers the variable y as well. We simply add to σ the assignment of the variable y to its best value v^* , thus obtaining $\sigma := \sigma_1 \cup \{y = v^*\}$. The best value of y is simply the value in the domain of y that is maximal w.r.t. the ordering $>_{\pi_i}$. Let Y be the ordered vector of the variables in the bucket, i.e. $Y := X_B^>$. Our problem then consists in finding a preferred solution of the bucket B and the lexicographical constraint $Y \leq_{lex} \sigma(Y)$.

If the bucket contained only a single constraint, then the problem would reduce to that of finding a support of the constraint that is worse than or equal to $\sigma(Y)$. We first test whether $\sigma(Y)$ satisfies the constraint. If not, we seek the next best tuple according to the lexicographical order. If the bucket contains multiple constraints, all defined on the same variables Y , then we can iterate this approach. Each time, we invoke a constraint with a given lexicographical upper bound σ , it will reduce this to a new upper bound. We iterate this approach until the given upper bound satisfies all the constraints or until we encounter a constraint that cannot be satisfied under the reduced bound. The solution obtained in the first case combines the supports of all the constraints of variable y . We therefore call it a *heavy support*. Whereas classic supports indicate that a single variable-value assignment is consistent w.r.t. a constraint, a heavy support indicates that a bucket has a solution under a given lexicographical bound.

A bucket can contain multiple constraints which have some variables in common, but not all. An example is

$$\begin{aligned} c_1 &:= (x_1 = 2 \wedge x_3 = 2 \wedge y = 1) \vee (x_1 = 2 \wedge x_3 = 1 \wedge y = 2) \\ c_2 &:= (x_1 = 2 \wedge x_2 = 2 \wedge y = 2) \vee (x_1 = 1 \wedge x_2 = 1 \wedge y = 1) \end{aligned}$$

Suppose that higher values are preferred for all variables and that the variables are ordered as follows: $x_1 > x_2 > x_3 > y$. If the initial bound is $\{x_1 = 2, x_2 = 2, x_3 = 2, y = 2\}$, we first invoke c_1 , which reduces this bound to $\{x_1 = 2, x_2 = 2, x_3 = 2, y = 1\}$. If the constraint c_2 ignores the variable x_3 , then it will return $\{x_1 = 1, x_2 = 1, y = 1\}$ as best support under $\{x_1 = 2, x_2 = 2, y = 1\}$. However, if x_3 is taken into account, then new bound should be $\{x_1 = 2, x_2 = 2, x_3 = 1, y = 2\}$. It is obtained by reducing the value for x_3 and by choosing the tuple $\{x_1 = 2, x_2 = 2, y = 2\}$ of c_2 . Hence, a constraint can modify the bound on a variable even if it does not contain it. We capture this by the following definition:

Definition 2 Let σ be an assignment to X and c be a constraint s.t. $X_c \subseteq X$. An assignment σ' to X is a support for c under σ iff (1) $\sigma' \leq_{lex} \sigma$ and (2) σ' satisfies c , i.e. $\sigma'[X_c] \in R_c$. An assignment σ' is a preferred support for c under σ iff σ' is a support for c under σ and there is no other support σ^* for c under σ s.t. $\sigma' <_{lex} \sigma^*$.

A preferred support for c under σ can be found by a generic algorithm. Customized algorithms for certain types of constraints (tables with forbidden tuples; lex constraints; numerical constraints) can speed up support finding and are a topic of future work.

Algorithm HEAVYSUPPORT(B, σ)

```

1.  $Q := \{c \in B \mid \sigma[X_c] \notin R_c\};$ 
2. while  $Q \neq \emptyset$  do
3.   choose best  $c$  from  $Q$ ;
4.   if  $c$  has a support under  $\sigma$  then
5.     let  $\sigma'$  be the preferred support of  $c$  under  $\sigma$ ;
6.     set  $\sigma$  to  $\sigma'$ ;
7.   else return  $\perp$ .
8.    $Q := \{c \in B \mid \sigma[X_c] \notin R_c\};$ 
9. return  $\sigma$ .

```

Figure 3. Algorithm for finding heavy supports.

An algorithm for finding a heavy support is described in figure 3. In each step, the algorithm determines the set of constraints that are not satisfied by the lexicographical bound. It selects such a constraint c , determines a preferred support for c , and uses it to reduce the lexicographical bound. Constraints having more important variables should be processed first as they can result in higher decreases of the bound. When reducing the bound, some variables are changing their value and all constraints containing those variables need to be rechecked. If they are not satisfied by the new bound, they are added to the queue Q . The process is repeated until the current bound satisfies all the constraints or there is a constraint that has no support under the given bound.

Theorem 2 *The algorithm HEAVYSUPPORT(B, σ) terminates after at most $O(d^{|Y|})$ iterations. If $B \cup \{Y \leq_{lex} \sigma(Y)\}$ is consistent, then it returns a solution of $\text{Lex}(Y)(B \cup \{Y \leq_{lex} \sigma(Y)\})$. Otherwise, it returns an inconsistency.*

It is important to understand the ‘backtracking’ behaviour of the algorithm. When seeking for a preferred support, the algorithm will first reduce the bound on the last variable. If this fails, it backtracks to the previous variable and will reduce its bound. By doing this, it can hit a ‘hole’, i.e. a variable that does not belong to the constraint. After reducing the bound on the ‘hole’, the algorithm can choose any value for the succeeding variables, return a support, and hand over control to a constraint having a variable covering the hole. However, it can also happen that there is no support given the values of the variables preceding the hole. In that case, the algorithm backjumps over the hole and changes a value of a variable that precedes the hole.

4 Conclusion

In [15], David McAllester has predicted that MAC-based solvers will outperform constraint engines based on truth maintenance. For more than one decade, this had been true. The recent success of conflict analysis and conflict learning in SAT solvers has changed the picture again. It became an intensive research topic to extend SAT by specific theories such as linear constraints, uninterpreted functions, and others, and to make conflict learning work for those problems. In this paper, we showed that conflict learning can profit from the structure of general CSPs as well. It reacts to the failure of synthesizing supports and the learned conflicts cut off whole subtrees and not just partial assignments. We thus obtain a smooth adaption of JTMS ideas from [6] to CSPs, while generalizing the form of conflicts and allowing compact representations of Craig interpolants in form of lexicographical constraints. The approach makes conflict learning more pervasive for CP and permits a return of the JTMS to CP. However, the approach may also be interesting for other applications of Craig interpolants [1, 16].

Our work has the same relation to BE as PrefAC [2] has to AC as it focuses effort to preferred values. We get the behaviour of PrefAC for problems with an induced width of 1. As LEXINTERPOLATE keeps results of subproblems that are solved again and again, it promises a better behaviour for proving inconsistencies as required by overconstrained configuration problems [11] and by certain software verification problems (such as the detection of tests that are always false). It also appears that we exchanged the roles of search and inference. Each constraint has a customized support finding algorithm, which is indeed a local backjumping [18] search algorithm. And LEXINTERPOLATE provides a generic algorithm that infers ‘fat’ conflicts based on the result of the local searches. It is important to note that this is only possible if the different subproblems share the same preference orders. We thus identified another case where preferences play an important role for problem solving [7].

Experimental evaluation for specific software verification problems is in progress. Future work will be devoted to incorporate the best-fail principle from [12] to deal with infinite domains as they occur in software verification problems.

REFERENCES

- [1] Eyal Amir and Sheila A. McIlraith, ‘Partition-based logical reasoning.’, in *KR 2000*, pp. 389–400, (2000).
- [2] Christian Bessière, Anaïs Fabre, and Ulrich Junker, ‘Propagate the right thing: how preferences can speed-up constraint solving’, in *IJCAI-03*, pp. 191–196, Acapulco, (2003).
- [3] Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen, ‘Logic programs with ordered disjunction’, *Computational Intelligence*, **20**, 335–357, (2004).
- [4] William Craig, ‘Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory’, *Journal of Symbolic Logic*, **22**, 269–285, (1957).
- [5] Rina Dechter, ‘Bucket elimination: A unifying framework for reasoning’, *Artificial Intelligence*, **113**(1-2), 41–85, (1999).
- [6] Jon Doyle, ‘A truth maintenance system’, *Artificial Intelligence*, **12**, 231–272, (1979).
- [7] Jon Doyle, ‘Prospects for preferences’, *Computational Intelligence*, **20**, 11–136, (2004).
- [8] Eugene C. Freuder, ‘Complexity of k-tree structured constraint satisfaction problems.’, in *AAAI-90*, pp. 4–9, (1990).
- [9] Matthew L. Ginsberg, ‘Dynamic backtracking’, *Journal of Artificial Intelligence Research*, **1**, 25–46, (1993).
- [10] Roberto J. Bayardo Jr. and Robert Schrag, ‘Using CSP look-back techniques to solve real-world SAT instances.’, in *AAAI-97*, pp. 203–208, (1997).
- [11] Ulrich Junker, ‘QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems’, in *AAAI-04*, pp. 167–172, (2004).
- [12] Ulrich Junker, ‘Preference-based inconsistency proving: When the failure of the best is sufficient’, in *IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, pp. 106–111, (2005).
- [13] Ulrich Junker, ‘Preference-based problem solving for constraint programming’, in *Preferences: Specification, Inference, Applications*, Dagstuhl Seminar Proceedings 04271, (2006).
- [14] Narendra Jussien, Romuald Debruyne, and Patrice Boizumault, ‘Maintaining arc-consistency within dynamic backtracking’, in *CP 2000*, pp. 249–261, (2000).
- [15] David A. McAllester, ‘Truth maintenance.’, in *AAAI-90*, pp. 1109–1116, (1990).
- [16] Kenneth L. McMillan, ‘Applications of Craig interpolants in model checking.’, in *TACAS 2005*, pp. 1–12, (2005).
- [17] Charles Petrie, ‘Revised dependency-directed backtracking for default reasoning’, in *AAAI-87*, pp. 167–172, (1987).
- [18] Patrick Prosser, ‘Hybrid algorithms for the constraint satisfaction problem’, *Computational Intelligence*, **9**, 268–299, (1993).
- [19] Daniel Sabin and Eugene C. Freuder, ‘Contradicting conventional wisdom in constraint satisfaction’, in *ECAI-94*, pp. 125–129, (1994).
- [20] João P. Marques Silva and Karem A. Sakallah, ‘GRASP: A search algorithm for propositional satisfiability’, *IEEE Trans. Computers*, **48**(5), 506–521, (1999).

Multi-Objective Propagation in Constraint Programming

Emma Rollon and Javier Larrosa¹

Abstract. *Bounding constraints* are used to bound the tolerance of solutions under certain undesirable features. Standard solvers propagate them one by one. Often times, it is easy to satisfy them independently, but difficult to satisfy them simultaneously. Therefore, the standard propagation methods fail. In this paper we propose a novel approach inspired in multi-objective optimization. We compute a *multi-objective lower bound set* that, if large enough, can be used to detect the inconsistency of the problem. Our experiments on two domains inspired in real-world problems show that propagation of additive bounding constraints using our approach is clearly superior than previous approaches.

1 Introduction

A *constraint satisfaction problem* (CSP) requires the assignment of a set of variables in such a way that a set of constraints is satisfied. Many problems have constraints that refer to different objectives. Consider the scheduling of a major football league. A good schedule should be *fair* (no team should feel that is treated *much* worse than any other,...), *exciting* (e.g., hot games should be scheduled in accordance to TV requests, several hot games should not take place the same day, ...), *safe* (e.g., rival teams should not play simultaneously in close stadiums, teams with violent followers should not play late in the evening or near a play-ground in the afternoon), etc. One way to deal with different criteria is to establish *tolerance bounds* for each one, and find satisfying assignments with respect to all of them. Often, these constraints are in conflict, meaning that assignments that are good with respect one constraint are likely to be bad with respect another. Typically, it is easy to satisfy each constraint independently, but it is hard to satisfy all of them simultaneously. Hence, the difficulty lays on the conjunction.

CSPs are usually solved by searching on the tree of possible assignments. After each assignment a *propagation* process takes place. Its goal is to detect if the current assignment can be extended to a solution. Most propagation algorithms detect and discard domain values that are inconsistent with the current assignment. If some variable loses all its values, the algorithm backtracks. Typically, each constraint is propagated independently. Namely, a value is removed if it is shown to be inconsistent with respect one of the constraints. The only *communication* between constraints is through *value pruning* (pruning one value due to one constraint, may produce the pruning of another value due to another constraint, yielding a cascade effect). This solving approach may not be strong enough for problems with conflicting bounding constraints. Consider the following two con-

straints over 0-1 variables:

$$x_1x_2 + x_2x_3 + x_3x_4 \geq 1; \quad \sum_{i=1}^4 x_i \leq 1$$

Note that every value is consistent with respect each constraint, because every value can be extended to satisfy it. However, it is clear that no value is consistent with respect the two simultaneously. This is a simple example that many solvers would deal efficiently with. However, we will show in Section 4 that, if constraints are more intricate, standard solvers may perform poorly.

We propose a more appropriate propagation method. Essentially, our approach consists on considering CSP subproblems as multi-objective minimization problems. The set of tolerance bounds plays the role of a multi-objective upper bound. Then, we compute a multi-objective lower bound that, if large enough, allows backtracking. More precisely, we use *multi-objective mini-bucket elimination* (MO-MBE) [10] as a general parameterizable multi-objective lower bound algorithm. Our experiments on two domains inspired on real world problems show the suitability of our approach.

2 Preliminaries

2.1 CSPs, Bounding Constraints, and MBE

A *constraint satisfaction problem* (CSP) is a triplet $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{R})$ where \mathcal{X} is an ordered set of *variables*. Each variable $x_i \in \mathcal{X}$ takes values on a finite *domain* $D_i \in \mathcal{D}$. \mathcal{R} is a set of *constraints*. For convenience, we will consider that a constraint $c \in \mathcal{R}$ is a boolean function over a subset of the variables $var(c) \subseteq \mathcal{X}$ called its *scope*. A tuple t is a (possibly partial) assignment over a subset of the variables $var(t) \subseteq \mathcal{X}$. A singleton tuple is noted $(x_i \leftarrow a)$ and the join of two tuples is noted $t \cdot t'$. If $var(t) = var(c)$, then $c(t)$ tells whether the assignment is allowed by the constraint. We allow partial assignments of functions. Thus, if $var(t) \subset var(c)$, then $c(t)$ is a new constraint in which variables of $var(t)$ have been *fixed* as indicated by t , and its new scope is $var(c) - var(t)$. If $var(t) \not\subseteq var(c)$, then $c(t)$ is the partial assignment of c with respect to $var(t) \cap var(c)$. A *solution* of a CSP is a complete assignment that satisfies all the constraints. Solving a CSP is in general NP-complete.

An *additive bounding constraint* is a pair (\mathcal{F}, \top) , where $f \in \mathcal{F}$ are cost functions and $\top \in \mathbb{N}$ is the maximum acceptable cost. For convenience, we define the sum of costs as $u \oplus v = \min\{u + v, \top\}$. Tuple t satisfies the constraint iff,

$$\sum_{f \in \mathcal{F}} f(t) < \top$$

In this paper we will consider CSPs with $p > 1$ additive bounding constraints. Note that we do not make any assumption over cost functions $f \in \mathcal{F}$, which makes the concept of additive bounding

¹ Universitat Politècnica de Catalunya, Spain, email: {erollon, larrosa}@lsi.upc.edu

```

function MBE( $z, \mathcal{F}, \top$ )
1. for each  $i = n..1$  do
2.    $\mathcal{B}_i := \{f \in \mathcal{F} \mid x_i \in \text{var}(f)\}$ 
3.    $\{\mathcal{P}_{i_1}, \dots, \mathcal{P}_{i_r}\} := \text{Partition}(z, \mathcal{B}_i);$ 
4.   for each  $k = 1..r$  do  $g_{i_k} := (\sum_{f \in \mathcal{P}_{i_k}} f) \downarrow x_i;$ 
5.    $F := (F \cup \{g_{i_1}, \dots, g_{i_r}\}) - \mathcal{B}_i;$ 
6. endfor
7. return  $g_1;$ 
endfunction

```

Figure 1. Mini-Bucket Elimination algorithm.

constraint extremely general. Besides, the ideas that we will introduce directly apply to more general bounding constraints such as *semiring-based* [2].

In recent years, many propagation algorithms for additive bounding constraints have been proposed. For instance, all *weighted CSP* local consistencies [9, 3] can be used for this purpose. Another alternative is *mini-bucket elimination* MBE [6], which is the basis of our work. The main advantage of MBE over local consistencies is its control parameter z that allows to trade resources for accuracy. For our purposes in this paper, MBE has the additional advantage of being extendible to multi-objective optimization. The main disadvantages are that MBE cannot be easily used for filtering [5] and that its implementation inside a solver is not incremental. MBE uses the following two operations over cost functions.

- Let f and g be two cost functions. Their sum, denoted $f + g$, is a new function with scope $\text{var}(f) \cup \text{var}(g)$ defined as,

$$(f + g)(t) = f(t) \oplus g(t)$$

- Let f be a cost function. The elimination of x_i , $f \downarrow x_i$, is a new function with scope $\text{var}(f) - \{x_i\}$ defined as,

$$f \downarrow x_i(t) = \min_{a \in D_i} \{f(t \cdot (x_i \leftarrow a))\}$$

Figure 1 shows MBE. It receives an additive bounding constraint (\mathcal{F}, \top) and returns a lower bound of the best assignment. MBE can be used as a propagation procedure inside search, because if the lower bound equals \top , the constraint cannot be satisfied and the current line of search can be abandoned. MBE processes variables in decreasing order. For each variable x_i , it computes \mathcal{B}_i the set of cost functions having x_i in the domain. \mathcal{B}_i is partitioned into subsets such that the join scope has size at most $z+1$. The functions of each subset are summed and variable x_i is eliminated. After the last elimination, a zero-arity (namely, a constant) contains the result. MBE is time and space exponential on parameter z .

2.2 Multi-Objective Minimization

Let us consider problems with p objectives. Let \vec{v} and \vec{u} be two distinct p -vectors. We say that \vec{v} dominates \vec{u} (noted $\vec{v} < \vec{u}$) if $\forall j, v_j \leq u_j$. A *multiobjective weighted constraint satisfaction problem* (MO-WCSP) is defined as $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \{\mathcal{F}_j, \top_j\}_{j=1}^p)$, where \mathcal{X} and \mathcal{D} are variables and domains. \mathcal{F}_j defines the j -th objective, $F_j(X) = \sum_{f \in \mathcal{F}_j} f(X)$. The multi-objective function is $F(X) = (F_1(X), \dots, F_p(X))$. Given a complete assignment X ,

we say that $F(X)$ is *consistent* iff $\forall j, F_j(X) < \top_j$. A *solution* is a consistent complete assignment. A solution X is *efficient* or *Pareto optimal* if there is no better solution (i.e., $\forall X', F(X') \not< F(X)$). \mathcal{X}_E is the set of efficient solutions and \mathcal{EF} is the corresponding *efficient frontier*. The task in a MO-WCSP is to compute \mathcal{EF} (and, possibly, one or all efficient solutions for each of its elements).

Multi-objective MBE (MO-MBE) [10] extends MBE to the multi-objective context. Let $\vec{\top} = (\top_1, \dots, \top_p)$. The sum of p -vectors is defined as,

$$\vec{v} + \vec{u} = \begin{cases} \vec{\top} & \exists j, v_j \oplus u_j = \top_j \\ (v_1 \oplus u_1, \dots, v_p \oplus u_p) & \text{otherwise.} \end{cases}$$

The non-domination closure of a set of p -vectors S is defined as $\langle S \rangle = \{v \in S \mid \forall u \in S, u \not< v\}$. MO-MBE works with p -functions which, instead of scalar costs, return sets of non-dominated p -vectors. For that purpose, original cost functions $f \in \mathcal{F}_j$ need to be reformulated from $f(t) = u$ to $f(t) = \{(0, \dots, 0, u, 0, \dots, 0)\}$ (where the u component is at the j -th position). Operations on functions are accordingly extended,

- Let f and g be p -functions. Their sum $h = f + g$ is defined as,

$$h(t) = \langle \{ \vec{v} \mid t = t' \cdot t'', \vec{v} = \vec{v}' + \vec{v}'', \vec{v}' \in f(t'), \vec{v}'' \in g(t'') \} \rangle$$
- Eliminating of x_i from a p -function f , $h = f \downarrow x_i$, is defined as,

$$h(t) = \langle \{ \vec{v} \mid \forall a \in D_i, \vec{v} \in f(t \cdot (x_i \leftarrow a)) \} \rangle$$

Note that, if $p = 1$, these definitions reduce to the classical ones.

Consider as an example the 2-functions f and g in Figure 2 with $\vec{\top} = (15, 18)$ under domains $\{a, b\}$. The sum $f + g$ is a 2-function $(f + g)(x_1, x_2, x_3)$. Note that in $(f + g)(a, b, a)$, the sum of the 2-vectors $(4, 10)$ and $(11, 1)$ is $\vec{\top}$. As $\vec{\top}$ is dominated by $(10, 12)$, it has been removed. The elimination of variable x_3 from $f + g$ is a 2-function $(f + g) \downarrow x_3(x_1, x_2)$. Note that in $(f + g) \downarrow x_3(a, a)$, the 2-vector $(4, 9)$ has been removed as a consequence of the non-domination closure. Moreover, $\vec{\top}$ has also been removed from $(f + g) \downarrow x_3(a, b)$ for the same reason.

The code of MO-MBE is the same as MBE in Figure 1. The only difference is that costs are now p -vectors, and cost functions are now p -functions. Given a MO-WCSP, MO-MBE returns a set S of non-dominated p -vectors which forms a *lower bound set* of its efficient frontier \mathcal{EF} (namely, $\forall \vec{v} \in \mathcal{EF}, \exists \vec{v}' \in S$ such that either $v = v'$ or $v' < v$). If MO-MBE returns $\vec{\top}$, the MO-WCSP does not have any solution. The time complexity of MO-MBE is $O(e \times \prod_{j=1}^{p-1} \top_j^2 \times d^z)$, where e is the number of p -functions and d is the largest domain size.

3 Propagating Additive Constraints using Multi-Objective Techniques

Let $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{R})$ be a CSP such that its set of constraints can be divided into two sets: \mathcal{H} is a set of arbitrary hard constraints, and $\{(\mathcal{F}_j, \top_j)\}_{j=1}^p$ are p additive bounding constraints. The problem is solved with a usual systematic search procedure. Consider an arbitrary search state with its associated partial assignment, which can be seen as if some of the domains have become singletons. Let $\mathcal{P}' = (\mathcal{X}', \mathcal{D}', \mathcal{R})$ be the associated subproblem. At this point, standard solvers would propagate the current assignment using general or specific algorithms for each type of constraint. Since we consider MBE as the propagation algorithm of bounding constraints and it does not prune unfeasible values, it makes sense to propagate in two

$f:$	$x_1 \quad x_2$	$\begin{array}{ c c c } \hline a & a & \{(3, 2), (2, 8)\} \\ a & b & \{(4, 10)\} \\ b & a & \{\top\} \\ b & b & \{\top\} \\ \hline \end{array}$	$f + g:$	$x_1 \quad x_2 \quad x_3$	$\begin{array}{ c c c c } \hline a & a & a & \{(4, 4), (3, 10)\} \\ a & a & b & \{(5, 3), (4, 9)\} \\ a & b & a & \{(10, 12)\} \\ a & b & b & \{\top\} \\ b & a & a & \{\top\} \\ b & a & b & \{\top\} \\ b & b & a & \{\top\} \\ b & b & b & \{\top\} \\ \hline \end{array}$	$(f + g) \downarrow x_3:$	$x_1 \quad x_2$	$\begin{array}{ c c c } \hline a & a & \{(4, 4), (3, 10), (5, 3)\} \\ a & b & \{(10, 12)\} \\ b & a & \{\top\} \\ b & b & \{\top\} \\ \hline \end{array}$
$g:$	$x_2 \quad x_3$	$\begin{array}{ c c c } \hline a & a & \{(1, 2)\} \\ a & b & \{(2, 1)\} \\ b & a & \{(6, 2), (11, 1)\} \\ b & b & \{\top\} \\ \hline \end{array}$						

Figure 2. Sum and projection over 2-functions. $\vec{T} = (15, 18)$.

steps: first the set \mathcal{H} , and second a sequence of MBE executions, one for each bounding constraint. The practical effectiveness of MBE can be greatly improved if each \mathcal{F}_j is augmented with the set of hard constraints \mathcal{H} ,

$$F_j(X) = \sum_{f \in \mathcal{F}_j \cup \mathcal{H}} f(X) < \top_j$$

For this purpose, boolean functions $c \in \mathcal{H}$ must be redefined in terms of costs (namely, $c(t) \in \{\top_j, 0\}$). Formally, we can see the propagation process as,

$$\text{Propagate}(\mathcal{H}) \wedge \bigwedge_{j=1}^p (\text{MBE}(z, \mathcal{F}_j \cup \mathcal{H}, \top_j) < \top_j)$$

If the previous expression returns *false* the search procedure should backtrack.

We propose the use of multi-objective mini-buckets instead of mono-objective mini-buckets. It requires the replacement of the sequence of calls to MBE by a single call to MO-MBE,

$$\text{Propagate}(\mathcal{H}) \wedge (\text{MO-MBE}(z, \{\mathcal{F}_j \cup \mathcal{H}, \top_j\}_{j=1}^p) < \vec{T})$$

The previous change may seem a minor modification. However, the subjacent algorithm is completely different and the kind of inference performed is much more powerful, as can be seen in the following example.

Consider a CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{R})$ with three 0/1 variables and two bounding constraints: $(F_1, 12)$ with $F_1 = \{f_1(x_1) = 10x_1, f_2(x_2) = 10x_2, f_3(x_3) = 2x_3\}$, and $(F_2, 10)$ with $F_2 = \{h_1(x_1) = 3(1 - x_1), h_2(x_2) = 4(1 - x_2), h_3(x_3) = 8(1 - x_3)\}$. There are two additional constraints $x_1 \neq x_2$ and $x_2 \vee x_3$. If we propagate with MBE($z = 2$) each bounding constraint (augmented with hard constraints) we obtain lower bounds 10 and 3, respectively. Find below the trace of each execution (note that in this example, there is no need to break buckets into mini-buckets),

1st bounding constr.	domain	value	g_3	g_2	$g_1()$
	1		0	2	10
2nd bounding constr.	0		2	10	
	1		0	4	3
	0		0	0	

The propagation indicates that the problem *may have* solution. Note that each lower bound is the optimum assignment of one bounding constraint discarding the other, so MBE is doing a perfect estimation with the information that it receives. Its problem is that it only *knows* part of the information. If we propagate with MO-MBE($z = 2$) the two bounding constraints simultaneously (augmented with hard constraints) we obtain a lower bound set of $\{\top\}$

which indicates that the problem *does not have* any solution. Find below the trace of the execution,

domain	value	g_3	g_2	$g_1()$
1		$\{(2, 0), (0, 8)\}$	$\{(2, 4)\}$	$\{\top\}$
0		$\{(2, 0)\}$	$\{(10, 8)\}$	

The key is that in each execution of MBE, it searches for one different valid assignment. Each one satisfies one bounding constraint separately, but not simultaneously. However, MO-MBE searches for a unique assignment that satisfies both bounding constraints simultaneously and cannot find it.

4 Computational Experiments

We tested our propagation mechanism in two different domains: *combinatorial auctions* and *satellite scheduling*. In both cases the problem contains two additive bounding constraints with unary cost functions plus a set of small arity hard constraints. We compare the performance of four algorithms based on depth-first search. The first is IlogSolver 6.1 with default parameters. Bounding constraints are encoded with the IloPack global constraint. The second algorithm enforces FDAC [9] with each bounding constraint augmented with the hard constraints. The third one enforces arc-consistency in the set of hard constraints and then executes MBE ($z = 2$) with the bounding constraints (namely, the first approach described in Section 3). The fourth one is like the third, but the two calls to MBE are replaced to one call to MO-MBE (namely, the second approach described in Section 3). For comparison, we always report cpu time. We run all the experiments on a Pentium IV at 3GHz with 2 GB of memory, running Linux.

4.1 Combinatorial Auctions

Combinatorial auctions (CA) allow bidders to bid for indivisible subsets of goods. In risk-conscious auctions, the bid-taker wants to control the risk of bid withdrawal following winner determination, because it may cause large losses in revenue [7]. Consider a set of goods $\{1, 2, \dots, n\}$ that go on auction. There are m bids. Bid i is defined by the subset of requested goods $X_i \subseteq \{1, 2, \dots, n\}$, the money offer b_i and the probability of payment failure r_i . The bid-taker must decide which bids are to be accepted. We consider a decision version of the problem in which the bid-taker provides two constants (P, R) indicating that she does not want to miss more than P over the maximum possible revenue ($\sum_{i \leq m} b_i$) and probability of bid withdrawal lower than R . Both constraints can be expressed as additive (the second one requires a probabilistic independence assumption and a logarithmic transformation).

We have generated mono-objective CA using the PATH model of the CATS generator [8] and randomly added payment failure probabilities to the bids in the range 0.0 to 0.3. We experiment on instances with 20 and 50 goods, varying the number of bids from 80 to 200. For each parameter configuration, we generate samples of size 25 and set the time limit to 300 seconds. For each instance, we established the values of (P, R) in such a way that *i*) the instance admits a solution and *ii*) a small decrease of either one renders the problem unsolvable. Consequently, the instances are difficult with respect to the two constraints.

Figure 3 reports the results for instances with 20 and 50 goods, respectively. It can be observed that problems become harder as the number of bids increases. Regarding the algorithms, it is clear that MO-MBE propagation always outperforms the other three approaches. For instances with 20 goods, it is about 6 times faster than its competitors. With 50 goods the gain is still larger (up to 10 times faster).

4.2 Earth observation satellite scheduling

An earth observation satellite, such as the Spot5 [1], orbits the earth while taking photographs requested by different customers. It is impossible to fulfill all the requests. Thus, the problem is to select the subset that the satellite will actually take and decide which camera will be used for each one. We experiment with Spot5 instances that have binary and ternary hard constraints and variables with domains of size 2 and 4. We consider instances with 2 bounding constraints. The first one comes from the on-board storage limit. The second one comes from the importance of photographs (each photograph has an associated penalty for not taking it). We consider the decision problem in which two constants (S, P) are given: the available on-board memory has size S and cannot be surpassed, and the maximum acceptable aggregated penalty is P . Since we could not solve complete instances, we considered subinstances as follows: $X_{\geq k}$ denotes instance X where photographs whose penalty is less than k have been eliminated.

Figure 4 reports the results for instance $1506_{\geq 1000}$. Since we observed that the behavior of other subinstances (i.e., $1401_{\geq 1000}$, $1403_{\geq 1000}$, $1405_{\geq 1000}$, and $1407_{\geq 1000}$) was very similar, we do not report their results. Each plot reports results for a fixed value of P and varying the value of S . We established a time limit of 600 seconds. Note the logarithmic scale. We observed that IlogSolver always performs very poorly and only solves instances with $S \leq 4$. Thus, we omit it from the plot.

Considering MBE and MO-MBE, we observe the following pattern that is best exemplified in the $P = 450000$ plot (Figure 4 top left). For high values of S , MBE is more efficient than MO-MBE. The reason is that the memory constraint is very easy to satisfy, which makes it practically irrelevant. MBE already captures the difficulty of the problem, which is mono-objective in nature. Thus, the higher overhead of MO-MBE is wasted. As the value of S decreases, the situation changes. Both bounding constraints become difficult to satisfy simultaneously. Propagating with mono-objective MBE fails in detecting inconsistency because it is easy to satisfy each constraint if the other one is disregarded, but it is difficult to satisfy the two of them simultaneously. Only the bi-objective nature of MO-MBE can capture such difficulty. As a result, MBE cannot solve the problems, while MO-MBE solves them in a few seconds. If S decreases even further, the memory constraint becomes clearly unsatisfiable in conjunction with the penalty constraint. MO-MBE propagation detects it easily but MBE propagation does not. Only for the lowest val-

ues of S , when the constraint is unsatisfiable independently of other constraints, MBE detects it efficiently. The algorithm that enforces FDAC behaves similarly to MBE because it also considers the two bounding constraints separately. However, it provides a much better average performance.

Observing the plots in decreasing order of P , we observe that problems become harder as the penalty bounding constraint becomes tighter and harder to satisfy. As before, there is a range of S for which the instances are most difficult. This difficulty peak shifts towards the right as P decreases. For MO-MBE propagation, the range is narrower than for MBE and FDAC, but it also fails to solve some instances within the time limit of 600 seconds.

The $P = 250000$ case requires further discussion: the plot only shows the left-hand side of the difficulty peak, where the tight memory constraint *helps* MO-MBE to prove unsatisfiability almost instantly whilst MBE and FDAC cannot. For large values of S the constraint becomes trivial and irrelevant. Then the problem difficulty is given only by the penalty constraint and the three algorithms fail in solving it.

5 Related Work

The idea of using the conjunction of two or more constraints during propagation, rather than using them one-by-one, is not new. For instance, path-consistency, path-inverse consistency and neighborhood inverse consistency [4] use this idea at different levels of sophistication. However, all these works assume binary problems and cannot be efficiently extended to higher arity constraints such as bounding constraints. The work of [12] is also related to ours. However, it is restricted to problems with so-called *knapsack constraints*, which are a special case of pairs of additive bounding constraints that share unary cost functions (namely, linear constraints of the form $L \leq AX \leq U$). A little bit more general is the work of [11], which applies to pairs of constraints of the form,

$$\sum_{i=1}^n w_i x_i \leq U \quad \wedge \quad \sum_{i=1}^n p_i x_i > U$$

Our notion of additive bounding constraint includes these and many other cases and allow us to take into account any number of bounding constraints. Besides, it can be easily extended to more sophisticated bounding constraints expressible in terms of semirings [2]. Moreover, our algorithmic approach using multi-objective optimization techniques is radically different.

6 Conclusions and Future Work

Additive bounding constraints, $\sum_{f \in \mathcal{F}} f(X) < \top$, are used to bound the tolerance under certain undesirable feature in problem solutions. The propagation in problems involving conflicting bounding constraints is a difficult task for standard solvers. Typically, they propagate constraints one by one. When it is easy to satisfy bounding constraints independently, but difficult to satisfy them simultaneously, this approach clearly fails. In this paper we have proposed a novel approach inspired in multi-objective optimization. We propagate the additive bounding constraints simultaneously with multi-objective mini-bucket elimination MO-MBE [10]. The output is a multi-objective lower bound set that can be used to detect the inconsistency of the problem. Our experiments on two domains inspired in real-world problems show that propagation of additive bounding constraints using MO-MBE is clearly superior than previous approaches.

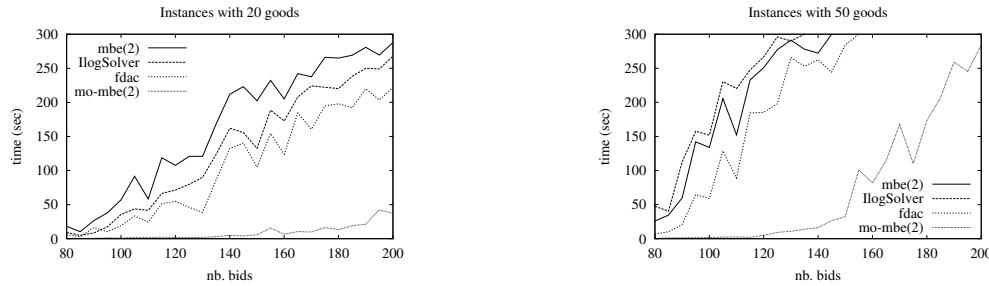


Figure 3. Experimental results on CA for 20 and 50 goods, respectively. Risk probabilities ranging from 0.0 to 0.3. Average time on 25 instances for each parameter configuration. Time limit 300 sec.

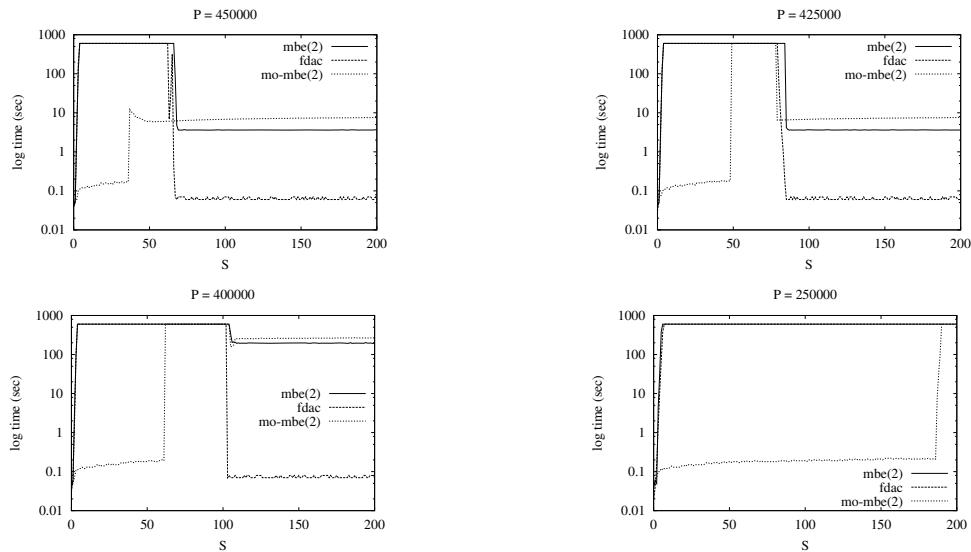


Figure 4. Experimental results on $1506_{\geq 1000}$ spot5 instance. Time limit 600 sec. Note the logarithmic scale.

The high overhead of multi-objective propagation may render it useless in problems with many bounding constraints. In that case, it may be useful to detect automatically pairs of conflicting constraints and apply MO-MBE to these pairs independently. Moreover, the experiments indicated that loose bounding constraints cause overhead but are of no use to our approach, so they should be detected and discarded in the propagation process. The development of this idea is part of our future work. A major drawback of MO-MBE propagation is that it cannot detect and prune unfeasible values. We want to overcome this problem using the ideas of [5].

ACKNOWLEDGEMENTS

This research has been funded with project TIN2005-09312-C03-02.

REFERENCES

- [1] E. Bensana, M. Lemaitre, and G. Verfaillie, ‘Earth observation satellite management’, *Constraints*, **4**(3), 293–299, (1999).
- [2] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie, ‘Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison’, *Constraints*, **4**, 199–240, (1999).
- [3] S. de Givry, F. Heras, J. Larrosa, and M. Zytnicki, ‘Existential arc consistency: getting closer to full arc consistency in weighted csp’s’, in *Proc. of the 19th IJCAI*, Edinburgh, U.K., (August 2005).
- [4] R. Debruyne and C. Bessière, ‘Domain filtering consistencies’, *Journal of Artificial Intelligence Research*, **14**, 205–230, (2001).
- [5] R. Dechter, K. Kask, and J. Larrosa, ‘A general scheme for multiple lower bound computation in constraint optimization’, in *CP-2001*, pp. 346–360, (2001).
- [6] R. Dechter and I. Rish, ‘Mini-buckets: A general scheme for bounded inference’, *Journal of the ACM*, **50**(2), 107–153, (March 2003).
- [7] A. Holland, *Risk Management for Combinatorial Auctions*, Ph.D. dissertation, Dept. of Computer Science, UCC, Ireland., 2005.
- [8] M. Pearson K. Leuton-Brown and Y. Shoham, ‘Towards a universal test suite for combinatorial auction algorithms’, *ACM E-Commerce*, 66–76, (2000).
- [9] J. Larrosa and T. Schiex, ‘In the quest of the best form of local consistency for weighted csp’, in *Proc. of the 18th IJCAI*, Acapulco, Mexico, (August 2003).
- [10] E. Rollon and J. Larrosa, ‘Depth-first mini-bucket elimination’, in *Proc. of the 11th CP*, pp. 563–577, Sitges (Spain), (2005). LNCS 3709.
- [11] Meinolf Sellmann, ‘Approximated consistency for knapsack constraints’, in *CP 2003*, pp. 679–693. LNCS 2833. Springer Verlag.
- [12] Michael Trick, ‘A dynamic programming approach for consistency and propagation for knapsack constraints’, *Annals of Op. Research*, **118**(118), 73–84, (2003).

Last Conflict based Reasoning

Christophe Lecoutre and Lakhdar Sais and Sébastien Tabary and Vincent Vidal¹

Abstract. In this paper, we propose an approach to guide search to sources of conflicts. The principle is the following: the last variable involved in the last conflict is selected in priority, as long as the constraint network can not be made consistent, in order to find the (most recent) culprit variable, following the current partial instantiation from the leaf to the root of the search tree. In other words, the variable ordering heuristic is violated, until a backtrack to the culprit variable occurs and a singleton consistent value is found. Consequently, this way of reasoning can easily be grafted to many search algorithms and represents an original way to avoid thrashing. Experiments over a wide range of benchmarks demonstrate the effectiveness of this approach.

1 Introduction

For solving instances of the Constraint Satisfaction Problem (CSP), tree search algorithms are commonly used. To limit their combinatorial explosion, various improvements have been proposed. Such improvements mainly concern ordering heuristics, filtering techniques and conflict analysis, and can be conveniently classified as look-ahead and look-back schemes [8]. *Look-ahead* schemes are used when the current partial solution has to be extended and *Look-back* schemes are designed to manage encountered dead-ends.

Early in the 90's, the look-ahead Forward-Checking (FC) algorithm [12] combined with the look-back Conflict-directed Back-Jumping (CBJ) technique [20] was considered as the most efficient approach to solve CSP instances. However, some years later, the look-ahead MAC algorithm, which maintains arc-consistency during search, was shown to be more efficient than FC-CBJ [21, 2].

Then, it became unclear if both paradigms were orthogonal, i.e. counterproductive one to the other, or not. Indeed, while it is confirmed by theoretical results [6] that the more advanced the forward phase is, the less useful the backward phase is, some experiments on hard, structured problems show that adding CBJ to M(G)AC can still present significant improvements. Further, refining the look-back scheme [11, 1, 16] by associating a so-called eliminating explanation (or conflict) with any value rather than with any variable gives to the search algorithm a more powerful backjumping ability. The empirical results in [1, 16] show that MAC can be outperformed by algorithms embedding such look-back techniques.

More recently, look-ahead techniques have taken an advantage with the introduction of conflict-directed variable ordering heuristics [4]. The idea is to associate a weight with any constraint C and to increment the weight of C whenever C is violated during search. As search progresses, the weight of hard constraints become more and more important, and this particularly helps the heuristic to select variables appearing in the hard part of the network. It does respect the

fail-first principle : "To succeed, try first where you are most likely to fail" [12]. The new conflict-directed heuristic $dom/wdeg$ is a very simple way to prevent thrashing [4, 13] and is an efficient alternative to backjump-based techniques [18].

Even if an advanced look-ahead technique is used, one can be interested by looking for the reason of an encountered dead-end as finding the ideal ordering of variables is intractable in practice. A dead-end corresponds to a conflict between a subset of decisions (variable assignments) performed so far. In fact, it is relevant (to prevent thrashing) to identify the most recent decision (let us call it the culprit one) that participates to the conflict. Indeed, once the culprit has been identified, we know that it is possible to safely backtrack up to it – this is the role of look-back techniques such as CBJ and DBT (Dynamic Backtracking) [11].

In this paper, we propose an original approach to (indirectly) backtrack to the culprit of the last encountered dead-end. To achieve it, the leaf conflict variable becomes in priority the next variable to be selected as long as the successive assignments that involves it render the network arc inconsistent. It then corresponds to checking the singleton consistency of this variable from the leaf toward the root of the search tree until a singleton value is found. In other words, the variable ordering heuristic is violated, until a backtrack to the culprit variable occurs and a singleton value is found.

In summary, the approach that we propose aims at guiding search to dynamically detect the conflict reason of the last encountered dead-end. It is important to remark that, contrary to sophisticated backjump techniques, our approach can be grafted in a very simple way to a tree search algorithm without any additional data structure.

2 Technical background

A Constraint Network (CN) P is a pair $(\mathcal{X}, \mathcal{C})$ where \mathcal{X} is a finite set of n variables and \mathcal{C} a finite set of e constraints. Each variable $X \in \mathcal{X}$ has an associated domain, denoted $dom(X)$, which contains the set of values allowed for X . Each constraint $C \in \mathcal{C}$ involves a subset of variables of \mathcal{X} , called scope and denoted $vars(C)$, and has an associated relation, denoted $rel(C)$, which contains the set of tuples allowed for its variables. A solution to a CN is an assignment of values to all the variables such that all the constraints are satisfied. A CN is said to be satisfiable iff it admits at least one solution. The Constraint Satisfaction Problem (CSP) is the NP-complete task of determining whether a given CN is satisfiable. A CSP instance is then defined by a CN, and solving it involves either finding one (or more) solution or determining its unsatisfiability. To solve a CSP instance, one can modify the CN by using inference or search methods [8]. Usually, domains of variables are reduced by removing inconsistent values, i.e. values that cannot occur in any solution. Indeed, it is possible to filter domains by considering some properties of constraint networks. Arc Consistency (AC) remains the central one.

¹ CRIL-CNRS FRE 2499, rue de l'université, SP 16, 62307 Lens cedex, France. email: {lecoutre,sais,tabary,vidal}@cril.univ-artois.fr

Definition 1 Let $P = (\mathcal{X}, \mathcal{C})$ be a CN. A pair (X, v) , with $X \in \mathcal{X}$ and $v \in \text{dom}(X)$, is arc consistent iff $\forall C \in \mathcal{C} \mid X \in \text{vars}(C)$, there exists a support of (X, v) in C , i.e., a tuple $t \in \text{rel}(C)$ such that $t[X] = v$ and $t[Y] \in \text{dom}(Y) \forall Y \in \text{vars}(C)$. P is arc consistent iff $\forall X \in \mathcal{X}$, $\text{dom}(X) \neq \emptyset$ and $\forall v \in \text{dom}(X)$, (X, v) is arc consistent.

Singleton Arc Consistency (SAC) [7] is a stronger consistency than AC, i.e. SAC can identify more inconsistent values than AC can. SAC guarantees that enforcing arc consistency after performing any variable assignment does not show unsatisfiability, i.e., does not entail a domain wipe-out.

Backtracking Search Algorithms The backtracking algorithm (BT) is a central algorithm for solving CSP instances. It employs a depth-first search in order to instantiate variables and a backtrack mechanism when dead-ends occur. Many works have been devoted to improve its forward and backward phases by introducing look-ahead and look-back schemes [8].

MAC [21] is the look-ahead algorithm which is considered as the most efficient generic approach to solve CSP instances. It simply maintains (re-establishes) arc consistency after each variable assignment. A dead-end is encountered if the network becomes arc inconsistent (because a domain is wiped out, i.e. becomes empty). When mentioning MAC, it is important to indicate which branching scheme is employed. Indeed, it is possible to consider binary (2-way) branching or non binary (d -way) branching. These two schemes are not equivalent as it has been shown that binary branching is more powerful (to refute unsatisfiable instances) than non-binary branching [14]. With binary branching, at each step of the search, a pair (X, v) is selected where X is an unassigned variable and v a value in $\text{dom}(X)$, and two cases are considered: the assignment $X = v$ and the refutation $X \neq v$. The MAC algorithm (using binary branching) can then be seen as building a binary tree. During search (i.e. when the tree is being built), we can make the difference between an opened node, for which only one case has been considered, and a closed node, for which both cases have been considered. Classically, MAC always starts by assigning variables before refuting values.

On the other hand, three representative look-back algorithms are SBT (Standard Backtracking), CBJ (Conflict Directed Backjumping) [20] and DBT (Dynamic Backtracking) [11]. The principle of these look-back algorithms is to jump back to a variable assignment that must be reconsidered as it is suspected to be the most recent culprit of the dead-end. SBT simply backtracks to the previously assigned variable whereas CBJ and DBT can identify a meaningful culprit decision by exploiting eliminating explanations.

Search Heuristics The order in which variables are assigned by a backtracking search algorithm has been recognized as a key issue for a long time. Using different variable ordering heuristics to solve a CSP instance can lead to drastically different results in terms of efficiency. In this paper, we will focus on the following representative variable ordering heuristics: dom , bz , dom/ddeg and dom/wdeg . The well-known dynamic heuristic dom [12] selects, at each step of the search, one variable with the smallest domain size. To break ties (which correspond to sets of variables that are considered as equivalent by the heuristic), one can use the current (also called dynamic) degree of the variable. This is the heuristic called bz [5]. By directly combining domain sizes and dynamic variable degrees, one obtains dom/ddeg [2] which can substantially improve the performance of the search on some problems. Finally, in [4], the heuristic

dom/wdeg has been introduced. The principle is to associate with any constraint of the problem a counter which is incremented whenever the constraint is involved in a dead-end. Hence, wdeg that refers to the weighted degree of a variable corresponds to the sum of the weights of the constraints involving this variable (and at least another unassigned variable).

3 Reasoning from conflicts

In this section, we show that it is possible to identify a nogood from a sequence of decisions leading to a conflict, and to exploit this nogood during search. We then discuss the impact of such a reasoning on thrashing prevention.

3.1 Nogood identification

Definition 2 Let $P = (\mathcal{X}, \mathcal{C})$ be a CN and (X, v) be a pair such that $X \in \mathcal{X}$ and $v \in \text{dom}(X)$. The assignment $X = v$ is called a positive decision whereas the refutation $X \neq v$ is called a negative decision. $\neg(X = v)$ denotes $X \neq v$ and $\neg(X \neq v)$ denotes $X = v$.

From now on, we will consider an inference operator ϕ assumed to satisfy some usual properties. This operator can be employed at any step of a tree search, denoted ϕ -search, using a binary branching scheme. For example, MAC corresponds to the ϕ_{AC} -search algorithm where ϕ_{AC} is the operator that establishes AC.

Definition 3 Let P be a CN and Δ be a set of decisions. $P|_{\Delta}$ is the CN obtained from P such that, for any positive decision $(X = v) \in \Delta$, $\text{dom}(X)$ is restricted to $\{v\}$, and, for any negative decision $(X \neq v) \in \Delta$, v is removed from $\text{dom}(X)$. $\phi(P)$ is the CN obtained after applying the inference operator ϕ on P .

If there exists a variable with an empty domain in $\phi(P)$ then P is clearly unsatisfiable, denoted $\phi(P) = \perp$.

Definition 4 Let P be a CN and Δ be a set of decisions. Δ is a nogood of P iff $P|_{\Delta}$ is unsatisfiable. Δ is a ϕ -nogood of P iff $\phi(P|_{\Delta}) = \perp$. Δ is a minimal ϕ -nogood of P iff $\nexists \Delta' \subset \Delta$ such that $\phi(P|_{\Delta'}) = \perp$.

Obviously, ϕ -nogoods are nogoods, but the opposite is not necessarily true. Our definition includes both positive and negative decisions as in [9, 17]. Note that we can consider a ϕ -nogood Δ as deduced from a sequence of decisions $\langle \delta_1, \dots, \delta_i \rangle$ such that $\Delta = \{\delta_1, \dots, \delta_i\}$. Such a sequence can correspond to the decisions taken along a branch in a search tree leading to a dead-end.

Definition 5 Let P be a CN and $\Sigma = \langle \delta_1, \dots, \delta_i \rangle$ be a sequence of decisions such that $\{\delta_1, \dots, \delta_i\}$ is a ϕ -nogood of P . A decision $\delta_j \in \Sigma$ is said to be culprit iff $\exists v \in \text{dom}(X_i) \mid \phi(P|_{\{\delta_1, \dots, \delta_{j-1}, \neg \delta_j, X_i=v\}}) \neq \perp$ where X_i is the variable involved in δ_i . We define the culprit subsequence of Σ to be either the empty sequence if no culprit decision exists, or the sequence $\langle \delta_1, \dots, \delta_j \rangle$ where δ_j is the latest culprit decision in Σ .

In other words, the culprit subsequence of a sequence of decisions Σ leading to an inconsistency ends in the most recent decision such that, when negated, there exists a value that can be assigned, without yielding an inconsistency with ϕ , to the variable involved in the last decision of Σ . We can show that it corresponds to a nogood.

Proposition 1 Let P be a CN and $\Sigma = \langle \delta_1, \dots, \delta_i \rangle$ be a sequence of decisions s.t. $\{\delta_1, \dots, \delta_i\}$ is a ϕ -nogood of P . The set of decisions contained in the culprit subsequence of Σ is a nogood of P .

Proof. Let $\langle \delta_1, \dots, \delta_j \rangle$ be the culprit subsequence of Σ . Let us demonstrate by recurrence that for any integer k such that $j \leq k \leq i$, the following hypothesis, denoted $H(k)$, holds:

$H(k)$: $\{\delta_1, \dots, \delta_k\}$ is a nogood

First, let us show that $H(i)$ holds. We know that $\{\delta_1, \dots, \delta_i\}$ is a nogood since, by hypothesis, $\{\delta_1, \dots, \delta_i\}$ is a ϕ -nogood of P . Then, let us show that, for $j < k \leq i$, if $H(k)$ holds then $H(k-1)$ also holds. As $k > j$ and $H(k)$ holds, we know that, by recurrence hypothesis, $\{\delta_1, \dots, \delta_{k-1}, \delta_k\}$ is a nogood. Furthermore, δ_k is not a culprit variable (since $k > j$). Hence, by Definition 5, we know that $\forall v \in \text{dom}(X_i), \phi(P|_{\{\delta_1, \dots, \delta_{k-1}, \neg\delta_k, X_i=v\}}) = \perp$. As a result, the set $\{\delta_1, \dots, \delta_{k-1}, \neg\delta_k\}$ is a nogood. By resolution, $\{\delta_1, \dots, \delta_{k-1}\}$ is a nogood. \square

The following property states that identifying an empty culprit subsequence allows proving the unsatisfiability of a CN.

Corollary 1 Let P be a CN and $\Sigma = \langle \delta_1, \dots, \delta_i \rangle$ be a sequence of decisions such that $\{\delta_1, \dots, \delta_i\}$ is a ϕ -nogood of P . If the culprit subsequence of Σ is empty, then P is unsatisfiable.

When we obtain a ϕ -nogood from a sequence of decisions Σ taken along a branch built by a ϕ -search algorithm, it is safe to backjump to the last decision contained in the culprit subsequence of Σ which corresponds to a nogood. We can also remark that the set of decisions contained in a culprit subsequence may not be a minimal nogood, and can be related to the nogood computed using the first Unique Implication Point (1UIP) scheme used in SAT solvers [22].

3.2 Last Conflict reasoning

The identification and exploitation of nogoods as described above can be easily embedded into a ϕ -search algorithm thanks to a simple modification of the variable ordering heuristic. We will call this approach *Last Conflict reasoning* (LC).

In practice, we will exploit LC only when a dead-end has been reached from an opened node of the search tree, that is to say, from a positive decision since when a binary branching scheme is used, positive decisions are generally taken first. It means that LC will be used iff δ_i (the last decision of the sequence mentioned in Definition 5) is a positive decision. To implement LC, it is then sufficient to (i) register the variable whose assignment to a given value directly leads to an inconsistency, and (ii) always prefer this variable in subsequent decisions (if it is still unassigned) over the choice given by the underlying heuristic – whatever heuristic is used. Notice that LC does not require any additional space cost.

Figure 1 illustrates *Last Conflict* reasoning. The leftmost branch on the figure corresponds to the positive decisions $X_1 = v_1, \dots, X_i = v_i$, such that $X_i = v_i$ leads to a conflict. At this point, X_i is registered by LC for future use. As a consequence, v_i is removed from $\text{dom}(X_i)$, i.e. $X_i \neq v_i$. Then, instead of pursuing the search with a new selected variable, X_i is chosen to be assigned with a new value v' . In our illustration, this leads once again to a conflict, v' is removed from $\text{dom}(X_i)$, and the process loops until all values are removed from $\text{dom}(X_i)$, leading to a domain wipe-out. The algorithm then backtracks to the assignment $X_{i-1} = v_{i-1}$, going to the right branch $X_{i-1} \neq v_{i-1}$. As X_i is still recorded by LC, it is then selected in priority, and all values of $\text{dom}(X_i)$ are excluded due to

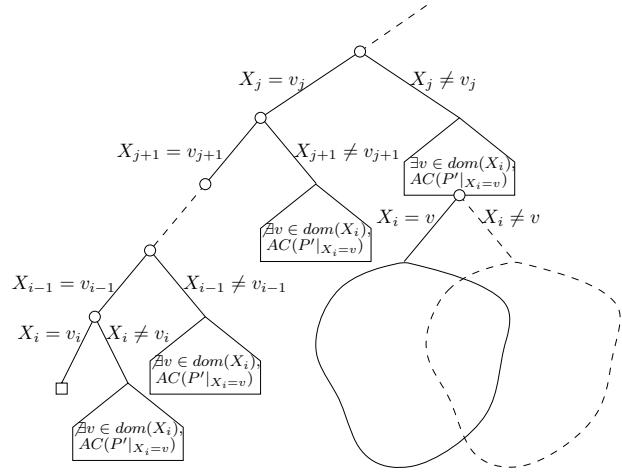


Figure 1. Last Conflict reasoning illustrated in a partial binary branching search tree. P' denotes the constraint network obtained at each node after performing the previous decisions and applying an inference operator ϕ .

the same process as above. The algorithm finally backtracks to the decision $X_j = v_j$, going to the right branch $X_j \neq v_j$. Then, as X_i is still the registered variable, it is preferred again and the values of $\text{dom}(X_i)$ are tested. But, as one of them (v) does not lead to a conflict, the search can continue with the assignment $X_i = v$. The variable X_i is then unregistered, and the choice for subsequent decisions is left to the underlying heuristic, until the next conflict occurs.

By using the ϕ_{AC} operator to identify culprit subsequences as described above, we obtain the following complexity results.

Proposition 2 Let $P = (\mathcal{X}, \mathcal{C})$ be a CN and $\Sigma = \langle \delta_1, \dots, \delta_i \rangle$ be a sequence of decisions such that $\{\delta_1, \dots, \delta_i\}$ is a ϕ_{AC} -nogood of P . The worst case time complexity of computing the culprit subsequence of Σ is $O(eid^3)$, where $e = |\mathcal{C}|$ and $d = |\text{dom}(X_i)|$ with X_i denoting the variable involved in δ_i .

Proof. The worst case is when the computed culprit subsequence of Σ is empty. In this case, it means that, for each decision, we check the singleton arc consistency of X_i . As checking the singleton arc consistency of a variable corresponds to d calls to an arc consistency algorithm, the worst case time complexity is id times the complexity of the used arc consistency algorithm. As the optimal worst case time complexity of establishing arc consistency is $O(ed^2)$ (e.g. AC2001/3.1 [3]), we obtain the overall time complexity $O(ed^3)$. \square

Corollary 2 Let $P = (\mathcal{X}, \mathcal{C})$ be a CN and $\Sigma = \langle \delta_1, \dots, \delta_i \rangle$ be a sequence of decisions that corresponds to a branch built by MAC leading to a failure. The worst case time complexity of computing the culprit subsequence of Σ is $O(\text{end}^3)$, where $n = |\mathcal{X}|$, $e = |\mathcal{C}|$ and $d = |\text{dom}(X_i)|$ with X_i denoting the variable involved in δ_i .

Proof. First, we know, that as positive decisions are performed first by MAC, the number of opened nodes in a branch of the search tree is at most n . Second, for each closed node, we do not have to check the singleton arc consistency of X_i since we have to directly backtrack. So we obtain $O(end^3)$. \square

3.3 Preventing thrashing using LC

Thrashing is the fact of repeatedly exploring the same subtrees. This phenomenon deserves to be carefully studied as an algorithm subject to thrashing can be very inefficient.

Sometimes, thrashing can be explained by some bad choices made earlier during search. In fact, whenever a value is removed from the domain of a variable, it can be explained (with more or less precision). It simply means that it is possible to indicate the decisions (i.e. variable assignments in our case) that entailed removing this value. By recording such so-called eliminating explanations and exploiting this information, one can hope to backtrack up to a level where a culprit variable will be re-assigned, this way, avoiding thrashing.

In some cases, no pertinent culprit variable(s) can be identified by a backjumping technique although thrashing occurs. For example, let us consider some unsatisfiable instances of the queens+knights problem as proposed in [4]. When the two subproblems are merged without any interaction (there is no constraint involving both a queen variable and a knight variable as in the *qk-25-25-5-add* instance), a backjumping technique such as CBJ or DBT is able to prove the unsatisfiability of the problem from the unsatisfiability of the knights subproblem (by backtracking up to the root of the search tree). When the two subproblems are merged with an interaction (queens and knights cannot be put on the same square as in the *qk-25-25-5-mul* instance), CBJ and DBT become subject to thrashing (when they are used with a standard variable ordering heuristic such as *dom*, *bz* and *dom/ddeg*) because the last assigned queen variable is considered as participating to the failure. The problem is that, even if there exists different eliminating explanations for a removed value, only the first one is recorded. One can still imagine to improve existing backjumping algorithms by updating eliminating explanations or computing new ones [15]. However, it is far beyond the scope of this paper.

Last Conflict reasoning is a new way of preventing thrashing, while still being a look-ahead technique. Indeed, guiding search to the last decision of a culprit subsequence behaves similarly to using a form of backjumping to that decision. For example, we can show that when a backjump to a culprit decision occurs with the Gaschnig's technique [10], then LC, in the same context, also reaches this decision in polynomial time.

Table 1 illustrates the powerful thrashing prevention capability of LC on the two instances mentioned above. SBT, CBJ and DBT cannot prevent thrashing for the *qk-25-25-5-mul* instance as, within 2 hours, the instance remains unsolved (even when other standard heuristics are used). On the other hand, in about 1 minute, LC (with SBT) can prove the unsatisfiability of this instance. The reason is that all knight variables are singleton arc inconsistent. When such a variable is reached, LC guides search up to the root of the search tree.

<i>Instance</i>		<i>SBT</i>	<i>CBJ</i>	<i>DBT</i>	<i>LC</i>
<i>qk-25-25-5-add</i>	<i>cpu</i>	> 2h	11.7	12.5	58.9
	<i>nodes</i>	—	703	691	10,053
<i>qk-25-25-5-mul</i>	<i>cpu</i>	> 2h	> 2h	> 2h	66.6
	<i>nodes</i>	—	—	—	9922

Table 1. Cost of running MAC-bz (time out set to 2 hours)

4 Experiments

In order to show the practical interest of the approach described in this paper, we have conducted an extensive experimentation (on a PC Pentium IV 2,4GHz 512Mo under Linux). Performances are measured in terms of the number of visited nodes (nodes) and the cpu time in seconds (cpu). Remark that for our experimentation, we have used MAC (with SBT, i.e. chronological backtracking), and studied the impact of LC wrt various variable ordering heuristics.

First, we have experimented different series of problems. The results that we have obtained are given in Table 2. The series corresponding to composed random instances, random 3-SAT instances

<i>dom/ddeg</i>		<i>dom/wdeg</i>		<i>bz</i>	
$\neg LC$	<i>LC</i>	$\neg LC$	<i>LC</i>	$\neg LC$	<i>LC</i>
Composed random instances (10 instances per series)					
25-1-40	3600 (10)	0.03	0.05	0.03	0.01
25-10-20	1789 (4)	0.32	0.09	0.08	1255 (3)
75-1-40	3600 (10)	0.10	0.10	0.90	0.02
Random 3-SAT instances (100 instances per series)					
<i>ehi-85</i>	1726	2.43	2.21	0.43	1236
<i>ehi-90</i>	3919	3.17	2.34	0.43	2440
Balanced QWH instances (100 instances per series)					
15-106	3.72	2.6	0.27	0.35	3.8
18-141	528 (4)	267 (1)	4.96	6.87	542 (4)
					274 (1)
Radar Surveillance instances (50 instances per series)					
<i>rs-24-2</i>	948 (13)	0.28	0.01	0.01	1989 (26)
<i>rs-30-0</i>	242 (3)	0.35	0.01	0.01	1108 (15)
					18

Table 2. Average cost (cpu) of running MAC without LC ($\neg LC$) and with LC on different series of problems

and balanced QWH instances were used as benchmarks² for the first CSP solver competition. Each composed random instance contains a main (under-constrained, here) fragment and some auxiliary fragments, each one being grafted to the main fragment by introducing some binary constraints. Each random 3-SAT instance embeds a small unsatisfiable part and has been converted to CSP using the dual encoding method. Each balanced QWH instance corresponds to a satisfiable balanced quasi-group instance with holes. Finally, the last series correspond to some realistic radar surveillance instances³ as proposed by the Swedish Institute of Computer Science (SICS). The problem is to adjust the signal strength (from 0 to 3) of a given number of fixed radars wrt 6 geographic sectors. Moreover, each cell of the geographic area must be covered exactly by 3 radar stations, except for some insignificant cells that must not be covered. Each set is denoted *rs-i-j* where *i* and *j* represent the number of radars and the number of insignificant cells, respectively.

In Table 2, we can observe the impact of LC when using MAC and different heuristics. Note that the time limit was fixed to 1 hour (except for the random 3-SAT instances, all solved in reasonable time) and that the number of expired instances, i.e. instances not solved within 1 hour of allowed cpu time, is given between brackets. Also, remark that, in case of expired instances, the indicated cpu must be considered as a lower bound.

On the one hand, when standard heuristics *dom/ddeg* and *bz* are used, it clearly appears that LC allows improving the efficiency of MAC in both CPU time and the number of solved instances, especially on composed and radar surveillance instances. In fact, these instances have a structure and a too blind search is subject to thrashing. Using LC avoids this phenomena without disturbing the main behaviour of the heuristics. On the other hand, when the conflict-directed heuristic *dom/wdeg* is used, LC is not so important since thrashing was already prevented by the heuristic.

Finally, we present in Table 3 some representative results obtained for some selected instances of the first international CSP competition. The time limit was also fixed to 1 hour. Once again, it appears that using LC with a standard heuristic greatly improves the efficiency of MAC. This is not always the case for MAC-*dom/wdeg*. Note that most of these instances cannot be efficiently solved using a backjumping technique such as CBJ or DBT combined with a standard heuristic as shown in [18].

² <http://cpai.ucc.ie/05/Benchmarks.html>

³ <http://www.ps.uni-sb.de/~walser/radar/radar.html>

		dom/ddeg		dom/wdeg		dom		bz	
		$\neg LC$	LC	$\neg LC$	LC	$\neg LC$	LC	$\neg LC$	LC
Academic instances									
<i>Golomb-11-sat</i>	<i>cpu nodes</i>	438 55, 864	146 19, 204	584 51, 055	149 12, 841	379 58, 993	134 21, 858	439 60, 352	147 21, 262
<i>BlackHole-4-4-0010</i>	<i>cpu nodes</i>	3.89 6, 141	3.60 5, 573	3.01 6, 293	5.26 9, 166	> 1h —	36.45 162K	> 1h —	28.50 106K
<i>cc-10-10-2</i>	<i>cpu nodes</i>	1238 543K	35.63 14, 656	3.10 2, 983	4.11 3, 526	99.95 180K	8.45 9, 693	1239 543K	35.50 14, 656
<i>qcp-20-balanced-23</i>	<i>cpu nodes</i>	> 1h —	201 141K	17.11 19, 835	1.00 1, 210	26.70 100K	2.62 6, 606	1.11 431	3.39 3, 470
<i>qk-25-25-5-add</i>	<i>cpu nodes</i>	> 1h —	57.30 10, 052	135 24, 502	63.64 11, 310	> 1h —	57.72 10, 053	> 1h —	57.35 10, 053
<i>qk-25-25-5-mul</i>	<i>cpu nodes</i>	> 1h —	66.34 9, 922	134 22, 598	68.31 9, 908	> 1h —	65.64 9, 922	> 1h —	66.23 9, 922
Real-world instances									
<i>e0ddr1-10</i>	<i>cpu nodes</i>	> 1h —	87.47 157K	18.85 37, 515	30.49 56, 412	102 307K	1.06 1, 390	> 1h —	52.11 94, 213
<i>enddr1-1</i>	<i>cpu nodes</i>	> 1h —	1.35 2, 269	0.72 1, 239	0.57 733	0.20 50	0.20 50	> 1h —	0.78 1, 117
<i>graph2-f25</i>	<i>cpu nodes</i>	> 1h —	77.86 54, 255	29.10 31, 492	3.53 3, 140	> 1h —	344 436K	> 1h —	72.23 51, 246
<i>graph8-f11</i>	<i>cpu nodes</i>	> 1h —	5.00 1, 893	7.54 5, 075	0.49 152	> 1h —	57.73 41, 646	> 1h —	49.58 22, 424
<i>scen11</i>	<i>cpu nodes</i>	95.84 31, 81	1.65 905	0.97 911	0.96 936	> 1h —	1104 804K	> 1h —	2942 1672K
<i>scen6-w2</i>	<i>cpu nodes</i>	> 1h —	0.45 405	0.51 741	0.29 272	> 1h —	0.51 706	> 1h —	0.46 318

Table 3. Cost of running MAC without LC ($\neg LC$) and with LC on academic and real-world instances

5 Conclusion

In this paper, we have introduced an original approach that can be grafted to any search algorithm based on a depth-first exploration. The principle is to select in priority the variable involved in the last conflict (i.e. the last assignment that failed) as long as the network cannot be made consistent. This way of reasoning allows preventing thrashing by backtracking to the most recent culprit of the last conflict. It can be done without any additional cost in space and with a worst-case time complexity in $O(end^3)$. Our extensive experimentation clearly shows the interest of this approach.

In our approach, the variable ordering heuristic is violated, until a backtrack to the culprit variable occurs and a singleton consistent value is found. However, there is an alternative which is to not consider the found singleton consistent value as the next value to be assigned. In this case, the approach becomes a pure inference technique which corresponds to (partially) maintaining a singleton consistency (SAC, for example) on the variable involved in the last conflict. This would be related to the recent “quick shaving” technique [19].

Acknowledgments

This paper has been supported by the CNRS, the “Planevo” project and the “IUT de Lens”.

REFERENCES

- [1] F. Bacchus, ‘Extending Forward Checking’, in *Proceedings of CP’00*, pp. 35–51, (2000).
- [2] C. Bessière and J. Régin, ‘MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems’, in *Proceedings of CP’96*, pp. 61–75, (1996).
- [3] C. Bessière, J.C. Régin, R.H.C. Yap, and Y. Zhang, ‘An optimal coarse-grained arc consistency algorithm’, *Artificial Intelligence*, **165**(2), 165–185, (2005).
- [4] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, ‘Boosting systematic search by weighting constraints’, in *Proceedings of ECAI’04*, pp. 146–150, (2004).
- [5] D. Brelaz, ‘New methods to color the vertices of a graph’, *Communications of the ACM*, **22**, 251–256, (1979).
- [6] X. Chen and P. van Beek, ‘Conflict-directed backjumping revisited’, *Journal of Artificial Intelligence Research*, **14**, 53–81, (2001).
- [7] R. Debruyne and C. Bessière, ‘Some practical filtering techniques for the constraint satisfaction problem’, in *Proceedings of IJCAI’97*, pp. 412–417, (1997).
- [8] R. Dechter, *Constraint processing*, Morgan Kaufmann, 2003.
- [9] F. Focacci and M. Milano, ‘Global cut framework for removing symmetries’, in *Proceedings of CP’01*, pp. 77–92, (2001).
- [10] J. Gaschnig, ‘Performance measurement and analysis of search algorithms’, Technical Report CMU-CS-79-124, Carnegie Mellon, (1979).
- [11] M. Ginsberg, ‘Dynamic backtracking’, *Artificial Intelligence*, **1**, 25–46, (1993).
- [12] R.M. Haralick and G.L. Elliott, ‘Increasing tree search efficiency for constraint satisfaction problems’, *Artificial Intelligence*, **14**, 263–313, (1980).
- [13] T. Hulubei and B. O’Sullivan, ‘Search heuristics and heavy-tailed behaviour’, in *Proceedings of CP’05*, pp. 328–342, (2005).
- [14] J. Hwang and D.G. Mitchell, ‘2-way vs d-way branching for CSP’, in *Proceedings of CP’05*, pp. 343–357, (2005).
- [15] U. Junker, ‘QuickXplain: preferred explanations and relaxations for over-constrained problems’, in *Proceedings of AAAI’04*, pp. 167–172, (2004).
- [16] N. Jussien, R. Debruyne, and P. Boizumault, ‘Maintaining arc-consistency within dynamic backtracking’, in *Proceedings of CP’00*, pp. 249–261, (2000).
- [17] G. Katsirelos and F. Bacchus, ‘Generalized nogoods in csp’s’, in *Proceedings of AAAI’05*, pp. 390–396, (2005).
- [18] C. Lecoutre, F. Boussemart, and F. Hemery, ‘Backjump-based techniques versus conflict-directed heuristics’, in *Proceedings of ICTAI’04*, pp. 549–557, (2004).
- [19] O. Lhomme, ‘Quick shaving’, in *Proceedings of AAAI’05*, pp. 411–415, (2005).
- [20] P. Prosser, ‘Hybrid algorithms for the constraint satisfaction problems’, *Computational Intelligence*, **9**(3), 268–299, (1993).
- [21] D. Sabin and E. Freuder, ‘Contradicting conventional wisdom in constraint satisfaction’, in *Proceedings of CP’94*, pp. 10–20, (1994).
- [22] L. Zhang, C.F. Madigan, M.W. Moskewicz, and S. Malik, ‘Efficient conflict driven learning in a Boolean satisfiability solver’, in *Proceedings of ICCAD’01*, pp. 279–285, (2001).

Dynamic Orderings for AND/OR Branch-and-Bound Search in Graphical Models

Radu Marinescu and Rina Dechter¹

Abstract. *AND/OR search spaces* have recently been introduced as a unifying paradigm for advanced algorithmic schemes for graphical models. The main virtue of this representation is its sensitivity to the structure of the model, which can translate into exponential time savings for search algorithms. Since the variable selection can have a dramatic impact on search performance when solving optimization tasks, we introduce in this paper a new *dynamic* AND/OR Branch-and-Bound algorithmic framework which accommodates variable ordering heuristics. The efficiency of the dynamic AND/OR approach is demonstrated empirically in a variety of domains.

1 INTRODUCTION

Graphical models (e.g. constraint and belief networks) are a powerful representation framework for various automated reasoning tasks. These models use graphs to capture conditional independencies between variables, allowing a concise representation of the knowledge, as well as efficient graph-based query processing algorithms. *Constraint Optimization Problems* such as finding a solution that violates the least number of constraints or finding the most likely state of a belief network can be defined within this framework and they are typically tackled with either *search* or *inference* algorithms [9].

The AND/OR search space for graphical models [8] is a newly introduced framework for search that is sensitive to the independencies in the model, often resulting in exponentially reduced complexities. It is based on a pseudo-tree that captures independencies in the graphical model, resulting in a search tree exponential in the depth of the pseudo-tree, rather than in the number of variables.

The AND/OR Branch-and-Bound algorithm (AOBB) is a new search method that explores the AND/OR search tree for solving optimization tasks in graphical models [16]. If restricted to a static variable ordering, the algorithm was shown to outperform a static version of the traditional OR Branch-and-Bound. In practice however, variable selection can have a dramatic influence on search performance for solving constraint satisfaction and optimization tasks [9]. In the context of OR search spaces there exists a whole line of research that mitigates this problem and focuses on the practical use of variable ordering heuristics during search. The most powerful OR Branch-and-Bound solvers developed in the past years, such as those maintaining a form directional local consistency [6] or those guided by bounded tree-clustering inference [7] rely heavily on variable ordering heuristics in order to improve efficiency.

In this paper we introduce a collection of *dynamic* AND/OR Branch-and-Bound algorithms that extend AOOB by combining the AND/OR decomposition principle with variable ordering heuristics. There are three approaches to incorporating dynamic orderings into

AOBB. The first one improves AOOB by applying an independent semantic variable ordering heuristic whenever the partial order dictated by the static decomposition principle allows. The second, orthogonal approach gives priority to the semantic variable ordering heuristic and applies problem decomposition as a secondary principle. Since the structure of the problem may change dramatically during search we introduce a third approach that uses a dynamic decomposition method coupled with semantic variable ordering heuristics.

We apply the dynamic AND/OR Branch-and-Bound algorithms on two optimization tasks: solving Weighted CSPs [6] and pure 0-1 Integer Linear Programming problems [17]. We experiment with various random models and real-world scheduling and resource allocation problems. Our results show conclusively that the new dynamic AND/OR approach outperforms significantly the classic OR as well as the existing static AND/OR tree search algorithms.

2 BACKGROUND

A finite *Constraint Optimization Problem* (COP) is a six-tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \otimes, \Downarrow, Z \rangle$, where $\mathcal{X} = \{X_1, \dots, X_n\}$ is a set of variables, $\mathcal{D} = \{D_1, \dots, D_n\}$ is a set of finite domains and $\mathcal{F} = \{f_1, \dots, f_m\}$ is a set of constraints. Constraints can be either *soft* (cost functions) or *hard* (sets of allowed tuples). Without loss of generality we assume that hard constraints are represented as (bi-valued) cost functions. Allowed and forbidden tuples have cost 0 and ∞ , respectively. The scope of function f_i , denoted $\text{scope}(f_i) \subseteq \mathcal{X}$, is the set of arguments of f_i . The operators \otimes and \Downarrow are defined as follows: $\otimes_i f_i$ is a *combination operator*, $\otimes_i f_i \in \{\prod_i f_i, \sum_i f_i\}$ and $\Downarrow_Y f$ is an *elimination operator*, $\Downarrow_Y f \in \{\max_{S-Y} f, \min_{S-Y} f\}$, where S is the scope of function f and $Y \subseteq \mathcal{X}$. The scope of $\Downarrow_Y f$ is Y .

An optimization task is defined by $g(Z) = \Downarrow_Z \otimes_{i=1}^m f_i$, where $Z \subseteq \mathcal{X}$. A *global optimization* is the task of finding the best global cost, namely $Z = \emptyset$. For simplicity we will develop our work assuming a COP instance with *summation* and *minimization* as combination and elimination operators, and a global cost function defined by $f(\mathcal{X}) = \min_{\mathcal{X}} \sum_{i=1}^m f_i$.

The *constraint graph* of a COP instance has the variables \mathcal{X} as its nodes and an arc connects any two variables that appear in the scope of the same function.

3 AND/OR SEARCH TREES

The classical way to do search is to instantiate variables one at a time, following a static/dynamic variable ordering. In the simplest case, this process defines a search tree (called here OR search tree), whose nodes represent states in the space of partial assignments. The traditional search space does not capture independencies that appear in

¹ University of California, Irvine, USA, email: {radum,dechter}@ics.uci.edu

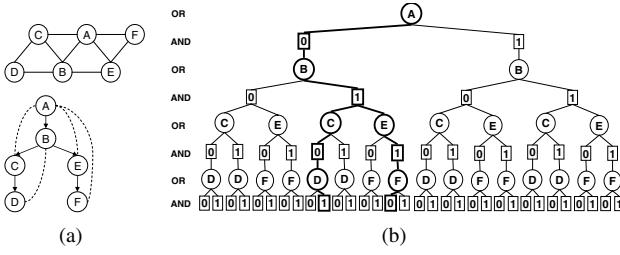


Figure 1. The AND/OR search space.

the structure of the constraint graph. Introducing AND states into the search space can capture the problem structure, decomposing the problem into independent subproblems by conditioning on values. The AND/OR search space is defined using a backbone *pseudo-tree* [10, 1, 8].

DEFINITION 1 (pseudo-tree) Given an undirected graph $G = (V, E)$, a directed rooted tree $T = (V, E')$ defined on all its nodes is called *pseudo-tree* if any arc of G which is not included in E' is a *back-arc*, namely it connects a node to an ancestor in T .

Given a COP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, its constraint graph G and a pseudo-tree T of G , the associated AND/OR search tree S_T has alternating levels of OR nodes and AND nodes. The OR nodes are labeled by X_i and correspond to the variables. The AND nodes are labeled by $\langle X_i, x_i \rangle$ and correspond to value assignments in the domains of the variables. The structure of the AND/OR tree is based on the underlying pseudo-tree T of G . The root of the AND/OR search tree is an OR node, labeled with the root of T .

The children of an OR node X_i are AND nodes labeled with value assignments $\langle X_i, x_i \rangle$, consistent along the path from the root, $path(X_i, x_i) = (\langle X_1, x_1 \rangle, \dots, \langle X_{i-1}, x_{i-1} \rangle)$. The children of an AND node $\langle X_i, x_i \rangle$ are OR nodes labeled with the children of variable X_i in T . In other words, the OR states represent alternative ways of solving the problem, whereas the AND states represent problem decomposition into independent subproblems, all of which need be solved. When the pseudo-tree is a chain, the AND/OR search tree coincides with the regular OR search tree.

A *solution subtree* Sol_{S_T} of S_T is an AND/OR subtree such that: (i) it contains the root of S_T ; (ii) if a nonterminal AND node $n \in S_T$ is in Sol_{S_T} then all its children are in Sol_{S_T} ; (iii) if a nonterminal OR node $n \in S_T$ is in Sol_T then exactly one of its children is in Sol_{S_T} .

Example 1 Figure 1(a) shows the constraint graph of a COP instance and a pseudo-tree together with the back-arcs (dotted lines). Figure 1(b) shows the AND/OR search tree based on the pseudo-tree, for bi-valued variables. A solution subtree is highlighted.

The AND/OR search tree can be traversed by a depth-first search algorithm that is guaranteed to have a time complexity exponential in the depth of the pseudo-tree and can use linear space [16]. The arcs from X_i to $\langle X_i, x_i \rangle$ are annotated by appropriate *labels* of the cost functions. The nodes in S_T can be associated with *values*, defined over the subtrees they root.

DEFINITION 2 (label) The label $l(X_i, x_i)$ of the arc from the OR node X_i to the AND node $\langle X_i, x_i \rangle$ is defined as the sum of all the cost functions values for which variable X_i is contained in their scope and whose scope is fully assigned along $path(X_i, x_i)$.

```

function: AOBB(st,  $\mathcal{X}$ ,  $\mathcal{D}$ ,  $\mathcal{F}$ )
1 if  $\mathcal{X} = \emptyset$  then return 0;
2 else
3    $X_i \leftarrow \text{SelectVar}(\mathcal{X})$ ;
4    $v(X_i) \leftarrow \infty$ ;
5   foreach  $x_i \in D_i$  do
6      $st' \leftarrow st \cup (X_i, x_i)$ ;
7      $h(X_i, x_i) \leftarrow \text{LB}(\mathcal{X}, \mathcal{D}, \mathcal{F})$ ;
8     foreach  $k = 1..q$  do
9        $h(X_k) \leftarrow \text{LB}(\mathcal{X}_k, \mathcal{D}_k, \mathcal{F}_k)$ ;
10       $\text{UpdateContext}(\text{out}, X_k, lb_k)$ ;
11    end
12    if  $\neg \text{FindCut}(X_i, x_i, \text{in}, \text{out}, h(X_i, x_i))$  then
13       $v(X_i, x_i) \leftarrow 0$ ;
14      foreach  $k = 1..q$  do
15         $\mathcal{D}'_k \leftarrow \text{LookAhead}(\mathcal{X}_k, \mathcal{D}_k, \mathcal{F}_k)$ ;
16        if  $\neg \text{EmptyDomain}(\mathcal{D}'_k)$  then
17           $val \leftarrow \text{AOBB}(st', \mathcal{X}_k, \mathcal{D}'_k, \mathcal{F}_k)$ ;
18           $v(X_i, x_i) \leftarrow v(X_i, x_i) + val$ ;
19        else
20           $v(X_i, x_i) \leftarrow \infty$ ;
21          break;
22        end
23         $v(X_i, x_i) \leftarrow v(X_i, x_i) + \text{label}(X_i, x_i)$ ;
24         $\text{UpdateContext}(\text{in}, v(X_i, x_i))$ ;
25         $v(X_i) \leftarrow \min(v(X_i), v(X_i, x_i))$ ;
26      end
27    end
28  return  $v(X_i)$ ;
29 end

```

Figure 2. AND/OR Branch-and-Bound search.

DEFINITION 3 (value) The value $v(n)$ of a node $n \in S_T$ is defined recursively as follows: (i) if $n = \langle X_i, x_i \rangle$ is a terminal AND node then $v(n) = l(X_i, x_i)$; (ii) if $n = \langle X_i, x_i \rangle$ is an internal AND node then $v(n) = l(X_i, x_i) + \sum_{n' \in \text{succ}(n)} v(n')$; (iii) if $n = X_i$ is an internal OR node then $v(n) = \min_{n' \in \text{succ}(n)} v(n')$, where $\text{succ}(n)$ are the children of n in S_T .

Clearly, the value of each node can be computed recursively, from leaves to root.

PROPOSITION 1 Given an AND/OR search tree S_T of a COP instance $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{F})$, the value $v(n)$ of a node $n \in S_T$ is the minimal cost solution to the subproblem rooted at n , subject to the current variable instantiation along the path from root to n . If n is the root of S_T , then $v(n)$ is the minimal cost solution to \mathcal{P} .

4 AND/OR BRANCH-AND-BOUND SEARCH

AND/OR Branch-and-Bound (AOBB) was recently proposed by [16] as a depth-first Branch-and-Bound that explores an AND/OR search tree for solving optimization tasks in graphical models. In this section we overview the static version of the algorithm.

4.1 Lower Bounds on Partial Trees

At any stage during search, a node n along the current path roots a current *partial solution subtree*, denoted by $G_{sol}(n)$, which must be connected, must contain its root n and will have a *frontier* containing all those nodes that were generated and not yet expanded. Furthermore, there exists a *static heuristic function* $h(n)$ underestimating $v(n)$ that can be computed efficiently when node n is first generated.

Given the current partially explored AND/OR search tree S_T , the *active path* $\mathcal{AP}(t)$ is the path of assignments from the root of S_T

to the current tip node t . The *inside context* $\text{in}(\mathcal{AP})$ of $\mathcal{AP}(t)$ contains all nodes that were fully evaluated and are children of nodes on $\mathcal{AP}(t)$. The *outside context* $\text{out}(\mathcal{AP})$ of $\mathcal{AP}(t)$, contains all the frontier nodes that are children of the nodes on $\mathcal{AP}(t)$. The *active partial subtree* $\mathcal{APT}(n)$ rooted at a node $n \in \mathcal{AP}(t)$ is the subtree of $G_{\text{sol}}(n)$ containing the nodes on $\mathcal{AP}(t)$ between n and t together with their OR children. A *dynamic heuristic evaluation function* of a node n relative to $\mathcal{APT}(n)$ which underestimates $v(n)$ is defined as follows (see [16] for more details).

DEFINITION 4 (dynamic heuristic evaluation function) *Given an active partial tree $\mathcal{APT}(n)$, the dynamic heuristic evaluation function of n , $f_h(n)$, is defined recursively as follows: (i) if $\mathcal{APT}(n)$ consists only of a single node n , and if $n \in \text{in}(\mathcal{AP})$ then $f_h(n) = v(n)$ else $f_h(n) = h(n)$; (ii) if $n = \langle X_i, x_i \rangle$ is an AND node, having OR children m_1, \dots, m_k then $f_h(n) = \max(h(n), l(X_i, x_i) + \sum_{i=1}^k f_h(m_i))$; (iii) if $n = X_i$ is an OR node, having an AND child m , then $f_h(n) = \max(h(n), f_h(m))$.*

4.2 Static AND/OR Branch-and-Bound

AOBB can calculate a *lower bound* on $v(n)$ for any node n on the active path, by using $f_h(n)$. It also maintains an *upper bound* on $v(n)$ which is the current minimal cost solution subtree rooted at n . If $f_h(n) \geq ub(n)$ then the search can be safely terminated below the tip node of the active path.

Figure 2 shows AOBB. The following notation is used: $(\mathcal{X}, \mathcal{D}, \mathcal{F})$ is the problem with which the procedure is called, st is the current partial solution subtree being explored, in (resp. out) represents the inside (resp. outside) context of the active path. These contexts are constantly updated during search. The algorithm assumes that variables are selected statically according to a pseudo-tree.

If the set \mathcal{X} is empty, then the result is trivially computed (line 1). Else, AOBB selects a variable X_i (i.e. expands the OR node X_i) and iterates over its values (line 5) to compute the OR value $v(X_i)$. For each value x_i the problem rooted by the AND node $\langle X_i, x_i \rangle$ is decomposed into a set of q independent subproblems $P_k = (\mathcal{X}_k, \mathcal{D}_k, \mathcal{F}_k)$, one for each child X_k of X_i in the pseudo-tree. Procedure LB computes the static heuristic function $h(n)$ for every node n in the search tree.

When expanding the AND node $\langle X_i, x_i \rangle$, AOBB successively computes the *dynamic heuristic evaluation function* $f_h(m)$ for every ancestor node m along the active path and terminates the current search path if, for some m , $f_h(m) \geq ub(m)$. Else, the independent subproblems are solved recursively (lines 14-22) and the results accumulated by the AND value $v(X_i, x_i)$ (line 18). A lookahead procedure is also executed in which unfeasible values are removed from future domains (line 15). After trying all feasible values of variable X_i , the minimal cost solution to the problem rooted by X_i remains in $v(X_i)$, which is returned (line 28).

5 DYNAMIC AND/OR BRANCH-AND-BOUND

In this section we go beyond static orderings and introduce a collection of *dynamic AND/OR Branch-and-Bound* algorithms that incorporate variable ordering heuristics used in the classic OR space. Similar structure-aware variable ordering heuristics have been previously investigated in the context of SAT and model counting [2, 11, 15].

We distinguish two classes of variable ordering heuristics. *Graph-based heuristics* (e.g. pseudo-tree arrangements) that try to maximize problem decomposition and *semantic-based heuristics* (e.g.

min-domain, max-degree, min-dom/deg) that aim at shrinking the search space. These two forces are orthogonal, namely we can use one as the primary goal and break ties based on the other. Moreover, we can use each class statically or dynamically. We present next three ways of combining efficiently these two classes of heuristics.

5.1 Partial Variable Ordering (PVO)

The first approach, called *AND/OR Branch-and-Bound with Partial Variable Ordering* (AOBB+PVO) combines the static graph-based decomposition given by a pseudo-tree with a dynamic semantic ordering heuristic. It is an adaptation of the ordering heuristics developed by [11] and [15] for solving large-scale SAT problem instances.

Let us illustrate the idea with an example. Consider the pseudo-tree from Figure 1(a) inducing the following variable group ordering: $\{A,B\}$, $\{C,D\}$, $\{E,F\}$; which dictates that variables $\{A,B\}$ should be considered before $\{C,D\}$ and $\{E,F\}$. Variables in each group can be dynamically ordered based on a second, independent semantic-based heuristic. Notice that after variables $\{A,B\}$ are instantiated, the problem decomposes into independent components that can be solved separately.

AOBB+PVO is similar to the algorithm from Figure 2 traversing an AND/OR search tree guided by a pre-computed pseudo-tree. Procedure `SelectVar` in this case implements the dynamic semantic ordering heuristic which selects next the best scoring variable from the current variable group of the pseudo-tree.

5.2 Dynamic Variable Ordering (DVO)

The second, orthogonal approach to PVO called *AND/OR Branch-and-Bound with Dynamic Variable Ordering* (DVO+AOBB), gives priority to the dynamic semantic ordering heuristic and applies static problem decomposition as a secondary principle during search. This idea was also explored by [2] in the context of SAT model counting.

DVO+AOBB is also based on the algorithm from Figure 2. It instantiates variables dynamically using a semantic ordering heuristic while constantly updating the problem graph structure. Specifically, after variable X_i is selected by procedure `SelectVar`, DVO+AOBB tentatively removes X_i from the graph and, if disconnected components are detected their corresponding subproblems are then solved separately and the results combined in an AND/OR manner (lines 14-22). It is easy to see that in this case a variable may have the best semantic heuristic to tighten the search space, yet, it may not yield a good decomposition for the remaining of the problem, in which case the algorithm would explore primarily an OR space.

5.3 Dynamic Separator Ordering (DSO)

The third approach, called *AND/OR Branch-and-Bound with Dynamic Separator Ordering* (AOBB+DSO), combines a dynamic graph-based decomposition heuristic with a dynamic semantic ordering heuristic giving priority to the first. The idea is supported by the constraint propagation (e.g. lookahead) procedure which may detect singleton variables (i.e. with only one feasible value left in their domains). When the value of a variable is known, we can remove it from the corresponding subproblem. Therefore, we may expect a better decomposition based on a simplified constraint graph.

AOBB+DSO is also based on the algorithm from Figure 2. It creates and maintains a separator S of the current constraint graph. A graph separator can be computed using the hypergraph partitioning method presented in [15]. The next variable is chosen dynamically

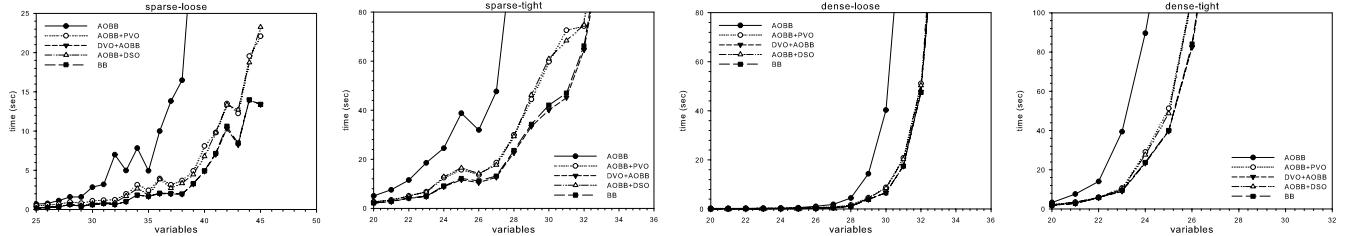


Figure 3. Time in seconds to prove optimality for random binary WCSPs.

from S by the semantic ordering heuristic until all variables are instantiated and the current problem decomposes into several subproblems. These independent subproblems are then solved separately and their solutions combined in an AND/OR fashion. The separator of each component is created from a simplified subgraph resulted from previous constraint propagation steps and it may differ for different value assignments. Clearly, if no singleton variables are discovered by the lookahead steps this approach is computationally identical to AOBB+PVO, although it may have a higher overhead due to the dynamic generation of the separators.

6 EXPERIMENTS

In this section we evaluate empirically the performance of the dynamic AND/OR Branch-and-Bound algorithms on two optimization tasks: solving Weighted CSPs and pure 0-1 Integer Linear Programming problems². For comparison, we include results obtained with the classic OR Branch-and-Bound (BB) algorithm.

BB explores an OR search tree in a depth-first manner maintaining an upper bound, the best solution cost found so far, and a heuristic evaluation function (i.e. lower bound) of the optimal extension of the current partial assignment. The subtree below the current path is pruned as soon as the lower bound exceeds the upper bound. The algorithm uses a dynamic semantic-based variable ordering heuristic.

Weighted CSP (WCSP) [6] extends the classic CSP formalism with so-called *soft constraints* which assign a positive integer penalty cost to each forbidden tuple (allowed tuples have cost 0). The goal is to find a complete assignment with minimum aggregated cost.

A *pure 0-1 Integer Linear Programming* (0-1 ILP) [17] consists of a set of integer decision variables (restricted to values 0 or 1) and a set of linear constraints (equalities or inequalities). The goal is to minimize a global linear cost function subject to the constraints.

The semantic-based variable ordering and heuristic evaluation function used in each domain by the OR and AND/OR Branch-and-Bound algorithms are as follows. For WCSPs we use the *min-dom/deg* heuristic (which selects the variable with the smallest ratio of the domain size divided by the future degree) and the heuristic evaluation function is computed by maintaining Existential Directional Arc Consistency (EDAC) [6]. For 0-1 ILPs we use the *fraction* heuristic (which selects the fractional variable closest to 0.5) and the heuristic evaluation function is computed by solving the linear relaxation of the current subproblem with the SIMPLEX³ method [5, 17]. Ties were broken lexicographically. The pseudo-tree that guides AOBB and AOBB+PVO as well as the graph separator used by AOBB+DSO were generated based on the recursive decomposition of a hypergraph representation of the problem [11, 15].

² All experiments were done on a 2.4GHz Pentium IV with 2GB of RAM.

³ Our code used the SIMPLEX implementation from the open source library

Table 1. Time in seconds to prove optimality for SPOT5 benchmarks.

spot	(w*,h)	BB time	AOBB time	AOBB+PVO time	DVO+AOBB time	AOBB+DSO time
29	(15, 22)	0.41	1.91	1.93	0.33	0.68
42	(38, 46)	-	-	-	4709	-
54	(12, 16)	0.06	1.13	1.07	0.03	0.08
404	(20, 25)	0.03	2.50	2.50	0.02	0.03
503	(11, 21)	11.70	2.78	2.78	0.03	0.13
505	(27, 42)	4010	75.43	75.37	17.28	82.74

We report the average CPU time required for proving optimality of the solution (the nodes visited are not shown for space reasons), the induced width (w^*) and depth of the pseudo-tree (h) obtained for the test instances. The best performance points are highlighted.

6.1 Random Weighted CSPs

We have generated 4 classes of binary random WCSPs with domain size and penalty weight set to 10 (Sparse-Loose, Sparse-Tight, Dense-Loose, Dense-Tight) as proposed in [6]. Figure 3 gives the time results in seconds (each data point represents an average over 20 samples). When comparing AOBB+PVO with static AOBB we notice a considerable improvement in terms of both running time and size of search space explored. AOBB+DSO has a similar performance as AOBB+PVO indicating that both algorithms use decompositions of similar quality. The best performance of all 4 problem classes is offered by DVO+AOBB and BB with no clear winner between the two. This implies that the semantic ordering heuristic is powerful and it does not leave much room for additional problem decompositions.

6.2 Earth Observing Satellites

The problem of scheduling an Earth observing satellite is to select from a set of candidate photographs, the best subset such that a set of imperative constraints are satisfied and the total importance of the selected photographs is maximized. We experimented with problem instances from the SPOT5 benchmark [3] which can be formulated as non-binary WCSPs [3]. For our purpose we considered a simplified MAX-CSP version of the problem where the goal is to minimize the number of imperative constraints violations.

Table 1 shows the results for 6 scheduling problems. We observe that in this domain DVO+AOBB is the best performing algorithm. In spot-42, the hardest instance, DVO+AOBB proves optimality in about one and a half hours, thus demonstrating the power of augmenting a semantic-based ordering heuristic with a static decomposition principle. The other algorithms exceeded the 10 hour time limit. AOBB+DSO is the second best algorithm, always exploring

lp_solve 5.5 available at http://groups.yahoo.com/group/lp_solve/

Table 2. Time in seconds to prove optimality for CELAR6 benchmarks.

celar6	(w*,h)	BB time	AOBB time	AOBB+PVO time	DVO+AOBB time	AOBB+DSO time
sub0	(7, 8)	0.88	0.73	0.81	0.92	0.71
sub1	(9, 9)	2260	3101	2200	2182	2246
sub1-24	(9, 9)	136	171	128	127	132
sub2	(10, 12)	4696	10963	7047	4024	6696
sub3	(10, 13)	14687	32439	28252	11131	28407
sub4-20	(11, 15)	681	137	157	70	179

fewer nodes than the other AND/OR algorithms. Its computational overhead of computing the separators dynamically did not pay off in some test cases (e.g. spot-505).

6.3 Radio Link Frequency Assignment Problem

RLFAP is a communication problem where the goal is to assign frequencies to a set of radio links in such a way that all links may operate together without noticeable interferences [4]. It can be naturally casted as a binary WCSP where each forbidden tuple has an associated penalty cost. Table 2 compares the OR and AND/OR algorithms for solving 6 publicly available RLFAP subinstances called CELAR6-SUB_i ($i = 1, \dots, 4$). We observe that in this domain also DVO+AOBB offers the best performance. On average, the speedups caused by DVO+AOBB over the other algorithms are as follows: 1.9x over AOBB, 1.6x over AOBB+PVO and 2.5x over BB. AOBB+DSO performs similarly to AOBB+PVO indicating that the quality of the dynamic problem decomposition is comparable to the static one.

6.4 Uncapacitated Warehouse Location Problem

In UWLP a company considers opening m warehouses at some candidate locations in order to supply its n existing stores. The objective is to determine which warehouse to open, and which of these warehouses should supply the various stores, such that the sum of the maintenance and supply costs is minimized. Each store must be supplied by exactly one warehouse. UWLP is typically formulated as a pure 0-1 integer linear program [17].

Table 3. Time in seconds to prove optimality for UWLP benchmarks.

uwlp	BB time	AOBB time	AOBB+PVO time	DVO+AOBB time
1	6.27	15.72	6.28	7.23
2	11.34	17.22	5.78	11.75
3	73.66	15.78	5.83	77.94
4	836.52	27.94	11.97	904.15
5	2501.75	32.69	16.98	2990.19
6	43.36	18.70	8.03	45.99
7	1328.40	27.89	8.53	1515.48
8	76.88	25.20	13.70	88.38
9	224.33	46.06	17.17	367.14
10	7737.65	28.03	9.13	9276.98

Table 3 displays the results obtained for 10 randomly generated UWLP problem instances⁴ with 50 warehouses and 200 stores. The warehouse opening and store supply costs were chosen uniformly randomly between 0 and 1000. These are large problems with 10,050 variables and 10,500 constraints. We can see that AOBB+PVO dominates in all test cases, outperforming BB (resp. DVO+AOBB) with several orders of magnitude in terms of running time and size of the search space explored. This is due to the problem's structure partially

⁴ We used the random problem generator from <http://www.mpi-sb.mpg.de/~units/ag1/projects/benchmarks/UfLib/>

captured by a shallow pseudo-tree with depth 123 and induced width 50. DVO+AOBB has a similar performance as BB (it is slower than BB due to its computational overhead), indicating that these problems do not break into disconnected components when semantic variable ordering has higher priority than problem decomposition.

In summary, for WCSP instances with relatively low connectivity (e.g. random WCSPs, SPOT5 networks) as well as for instances with higher graph density (e.g. CELAR6 networks) DVO+AOBB outperforms significantly both its OR and AND/OR competitors. For sparse 0-1 integer programs (e.g. UWLP) which yield extremely shallow pseudo-trees AOBB+PVO appears to be the preferred choice.

7 CONCLUSION

In this paper we extended the AND/OR Branch-and-Bound algorithmic framework with dynamic orderings for combining efficiently variable ordering heuristics with problem decomposition principles. The effectiveness of the dynamic AND/OR approach is demonstrated empirically in a variety of domains including random models as well as hard real-world problem instances.

Related Work: AOBB is related to the Branch-and-Bound method proposed by [13] for acyclic AND/OR graphs and game trees, as well as to the pseudo-tree search algorithm proposed in [14] for boosting Russian Doll search. BTD algorithm developed in [12] for semi-ring CSPs can also be interpreted as an AND/OR graph search algorithm.

ACKNOWLEDGEMENTS

This work was supported by the NSF grant IIS-0412854.

REFERENCES

- [1] R. Bayardo and D. Miranker, ‘On the space-time trade-off in solving constraint satisfaction problems.’, in *IJCAI*, pp. 558–562, (1995).
- [2] R. Bayardo and J. D. Pehoushek, ‘Counting models using connected components’, in *AAAI/IAAI*, pp. 157–162, (2000).
- [3] E. Bensana, M. Lemaître, and G. Verfaillie, ‘Earth observation satellite management.’, *Constraints*, 4(3), 293–299, (1999).
- [4] B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J. Warners, ‘Radio link frequency assignment.’, *Constraints*, 1(4), 79–89, (1999).
- [5] G.B. Dantzig, ‘Maximization of a linear function of variables subject to linear inequalities.’, *Activity Analysis of Production and Allocation*, (1951).
- [6] S. de Givry, F. Heras, J. Larrosa, and M. Zytnicki, ‘Existential arc consistency: getting closer to full arc consistency in weighted csp’s.’, in *IJCAI*, pp. 84–89, (2005).
- [7] R. Dechter, K. Kask, and J. Larrosa, ‘A general scheme for multiple lower bound computation in constraint optimization’, in *CP*, pp. 346–360, (2001).
- [8] R. Dechter and R. Mateescu, ‘Mixtures of deterministic-probabilistic networks.’, in *UAI*, pp. 120–129, (2004).
- [9] Rina Dechter, *Constraint Processing*, MIT Press, 2003.
- [10] E. Freuder and M. Quinn, ‘Taking advantage of stable sets of variables in constraint satisfaction problems.’, in *IJCAI*, pp. 1076–1078, (1985).
- [11] J. Huang and A. Darwiche, ‘A structure-based variable ordering heuristic.’, in *IJCAI*, pp. 1167–1172, (2003).
- [12] P. Jegou and C. Terrioux, ‘Decomposition and good recording for solving max-csp’s.’, in *ECAI*, (2004).
- [13] L. Kanal and V. Kumar, *Search in artificial intelligence.*, Springer-Verlag., 1988.
- [14] J. Larrosa, P. Meseguer, and M. Sanchez, ‘Pseudo-tree search with soft constraints.’, in *ECAI*, pp. 131–135, (2002).
- [15] W. Li and P. van Beek, ‘Guiding real-world sat solving with dynamic hypergraph separator decomposition.’, in *ICTAI*, pp. 542–548, (2004).
- [16] R. Marinescu and R. Dechter, ‘And/or branch-and-bound for graphical models.’, in *IJCAI*, pp. 224–229, (2005).
- [17] G. Nemhauser and L. Wolsey, *Integer and combinatorial optimization.*, Wiley, 1988.

Compact Representation of Sets of Binary Constraints

Jussi Rintanen¹

Abstract. We address the problem of representing big sets of binary constraints compactly. Binary constraints in the form of 2-literal clauses are ubiquitous in propositional formulae that represent real-world problems ranging from model-checking problems in computer-aided verification to AI planning problems. Current satisfiability and constraint solvers are applicable to very big problems, and in some cases the physical size of the problem representations prevents solving the problems, not their computational difficulty. Our work is motivated by this observation.

We propose graph-theoretic techniques based on cliques and bicliques for compactly representing big sets of binary constraints that have the form of 2-literal clauses. An n, m biclique in a graph associated with the constraints can be very compactly represented with only $n + m$ binary constraints and one auxiliary variable. Cliques in the graph are associated with *at-most-one* constraints, and can be represented with a logarithmic number of binary constraints. The clique representation turns out to be a special case of the biclique representation. We demonstrate the effectiveness of the biclique representation in making the representation of big planning problems practical.

1 Introduction

Binary constraints in the form of binary clauses $l_1 \vee l_2$ are very common in many applications of CSPs and propositional satisfiability. For example, most of the clauses in the representation of AI planning problems as CNF formulae in the propositional logic are binary clauses [11]. In structured formulae stemming from real-world applications, sets of binary clauses have regularities that make efficient reasoning and more compact representation possible. The goal of this work is to develop techniques for representing big sets of binary clauses more compactly.

The applications that motivate our work are planning and model-checking. The *planning as satisfiability* approach was introduced by Kautz and Selman [10, 11] and later generalized to a wide class of model-checking problems [2]. Current generation of satisfiability algorithms can solve planning and model-checking problems with thousands of variables and tens of thousands of actions. Interestingly, an obstacle to the scalability of the approach to still bigger problems is in some cases the enormous size of the propositional formulae.

For AI planning, for instance, there exist asymptotically optimal linear size translations into the propositional logic [14]. However, a part of the problem-specific information which is necessary for efficient reasoning, in the form of 2-literal *invariants* [3, 13], has a size that is in the worst-case quadratic in the size of the problem instance. Typically these 2-literal clauses constitute between 50 and 95 per cent of the formulae, or even more. An example of a typical invariant in a planning problem is for example $\neg\text{in}(A, \text{London}) \vee \neg\text{in}(A, \text{Paris})$

which indicates that an object cannot be in two locations simultaneously. For n state variables there are $\mathcal{O}(n^2)$ invariants that are 2-literal clauses. For a planning problem with $n = 5000$ state variables and a formula that encodes the search for plans of 100 time points, if only one of the state variables is true at any given time, the corresponding set of binary clauses has cardinality $100 \times \frac{n(n-1)}{2} = 1249750000 \sim 1.2 \times 10^9$. If each clause takes 10 bytes then the total size of the representation is about 12 gigabytes. The problem of representing the constraint graphs compactly arises.

In this paper we propose techniques for representing the constraint graphs more compactly with cliques and bicliques. We show how constraint graphs that contain big cliques or bicliques can be represented more compactly than the explicit representation, and we also show that constraint graphs arising from practically interesting applications indeed do contain big cliques and bicliques. The presence of cliques is more obvious and they have been addressed in earlier research. The main contribution of our work is the introduction of techniques based on bicliques. The underlying ideas in the use of bicliques are simple and powerful, yet their importance has not been recognized before. We show that the clique based techniques are subsumed by the biclique based techniques.

The outline of this paper is as follows. Section 2 introduces the graph-theoretic concepts of cliques and bicliques and discusses their computational properties. In Section 3 we discuss the compact representation of cliques of constraint graphs. In Section 4 we address the compact representation of constraint graphs more generally in terms of bicliques. In Section 5 we show the effectiveness of the biclique representation in reducing the sizes of formulae that represent an industrial size planning problem.

2 Preliminaries: Cliques and Bicliques

The techniques presented in this paper are based on cliques and bicliques of graphs.

Definition 1. Let $\langle N, E \rangle$ be an undirected graph. Then a clique is $C \subseteq N$ such that $\{n, n'\} \in E$ for every $n, n' \in C$ such that $n \neq n'$.

Hence cliques are complete subgraphs. Identification of cliques is expensive. Testing whether a graph contains a clique of size n is NP-complete [9, 6]. Hence we cannot expect to have a polynomial-time algorithm that is guaranteed to find the biggest cliques of a graph. Approximation algorithms for identifying cliques and related subgraphs exist [8]. Hochbaum's algorithms are based on reducing the problem to integer programming and showing that the corresponding linear programs have integers solutions. Linear programming problems can be solved in polynomial time.

Instead of using Hochbaum's algorithm, we have used the simpler polynomial-time algorithm without approximation guarantees that is given in Figure 1. In many of our example applications (see Section

¹ National ICT Australia, Canberra Research Laboratory, Canberra, Australia

5) this algorithm identifies all the maximal cliques because no two maximal cliques share a node. Of course, maximal cliques of many graphs do not have this property.

```

1: procedure partition-to-cliques( $N, E$ )
2:    $n := 1$ ;
3:    $S[1] := N$ ;
4:   change := true;
5:   while change do
6:     change := false;
7:     if there are  $l_1, l_2 \in S[i]$  for some  $i \in \{1, \dots, n\}$ 
8:       such that  $\{l_1, l_2\} \notin E$  and  $l_1 \neq l_2$  then
9:         change := true;
10:         $n := n + 1$ ;
11:         $S[n] := \{l_2\} \cup \{l | \{l, l_2\} \in E\}$ ;
12:         $S[i] := S[i] \setminus S[n]$ ;
13:      end if
14:   end while
```

Figure 1. A polynomial-time algorithm for partitioning the nodes of a graph to cliques. It identifies candidate cliques $S[i]$ that contain two nodes without an edge between them, and splits them to two new candidate cliques.

Theorem 2. *The algorithm partition-to-cliques with input $G = \langle N, E \rangle$ terminates after a number of execution steps that is polynomial in the size of G , and after termination $S[i]$ is a clique of G for every $i \in \{1, \dots, n\}$.*

Proof. Sketch: The loop can only exit if for every $i \in \{1, \dots, n\}$ there is an edge between every two nodes in $S[i]$. Hence on termination every $S[i]$ is a clique.

Since at every iteration n is incremented and the sets $S[1], \dots, S[n]$ form a partition of all literals to nonempty sets, the algorithm terminates at the latest when all $S[i]$ have size 1. Hence the number of literals is an upper bound on the number of iterations. On every iteration the amount of computation is polynomial. \square

A concept related to cliques is that of bicliques.

Definition 3. *Let $\langle N, E \rangle$ be an undirected graph. A biclique is a pair C, C' of sets $C \subseteq N$ and $C' \subseteq N$ such that $C \cap C' = \emptyset$ and $\{\{n_1, n_2\} | n_1 \in C, n_2 \in C'\} \subseteq E$.*

In other words, bicliques are complete bipartite subgraphs of a graph. The problem of testing for existence of bicliques under several size measures is NP-complete [16, 6, 12].

We have used a simple polynomial-time algorithm for identifying bicliques in constraint graphs. The algorithm is given in Figure 2. It

```

1: procedure identify-biclique( $N, E$ )
2:    $C_1 := \emptyset$ ;
3:    $C_2 := N$ ;      (*  $C_1, C_2$  is now a trivial 0-edge biclique *)
4:   repeat
5:      $n :=$  a node in  $N \setminus C_1$ ;          (* Maximizing... *)
6:           (*  $|C_2|$  for the new value of  $C_2$  if  $C_1 = \emptyset$  *)
7:           (*  $(|C_1| \times |C_2|) - (|C_1| + |C_2|)$  if  $C_1 \neq \emptyset$  *)
8:      $C_1 := C_1 \cup \{n\}$ 
9:      $C_2 := C_2 \cap \{m \in N \setminus C_1 | \{n, m\} \in E\}$ ;
10:   until  $|C_1| \times |C_2| - (|C_1| + |C_2|)$  does not increase;
```

Figure 2. A polynomial-time algorithm for finding bicliques

starts with the dummy biclique \emptyset, N , and repeatedly adds nodes to the first part. Nodes from the second part are removed if there is no edge between them and the new node in the first part. The nodes are chosen to maximize the value of the biclique that is being constructed (the size reduction obtained by the compact representation.) The algorithm terminates when the value of the biclique does not increase any more. The passage from \emptyset, N to $\{n\}, N'$ at the first step never increases the value of the biclique and this case is handled specially.

Theorem 4. *The algorithm identify-biclique with input $G = \langle N, E \rangle$ terminates after a number of execution steps that is polynomial in the size of G , and after termination C_1, C_2 is a biclique of G .*

Proof. First verify that after every iteration C_1, C_2 is a biclique.

The number of iterations is bounded by $|N|$ because after $|N| - 1$ iterations $C_1 = N \setminus \{m\}$ for some m and $C_2 \subseteq \{m\}$ and therefore $|C_1| \times |C_2| - (|C_1| + |C_2|)$ is $(|N| - 1) \times 1 - (|N| - 1 + 1) = -1$ or $(|N| - 1) \times 0 - (|N| - 1 - 0) = -|N| + 1$, and after $|N|$ iterations it is $|N| \times 0 - (|N| + 0) = -|N|$, and the iteration terminates. Polynomial runtime follows because computation on every iteration takes polynomial time. \square

3 Cliques in Constraint Graphs

We consider constraint graphs $\langle N, E \rangle$ that represent sets S of 2-literal clauses $l \vee l'$. The set N of nodes of the graph consists of all the literals a and $\neg a$ for $a \in A$ where A is the set of propositional variables, and there is an (undirected) edge $\{l, l'\} \in E$ between the nodes l and l' if and only if $l \vee l' \in S$ or $l' \vee l \in S$.

The clause set may have a very irregular structure and the best way of representing it may be as is, but in many cases there are regularities that make a more compact representation possible. The simplest regularity is perhaps the presence of an *at-most-one* constraint for a subset V of the variables, forbidding more than one of the variables to be true at a time. Hence for all $\{v, v'\} \subseteq V$ such that $v \neq v'$, the constraint graph has an edge $\{\neg v, \neg v'\} \in E$. The set of literals $\{\neg v | v \in V\}$ forms a clique in the constraint graph.

In the following sections we discuss ways of representing cliques and other regularities of constraint graphs more compactly. Irrespective of the details of the representation, the more compact representation of the cliques in a constraint graph proceeds in the same way.

1. Identify a big clique in the constraint graph.
2. If only small cliques were found, go to the last step.
3. Construct a propositional formula that represents the binary constraints corresponding to the clique more compactly (see Sections 3.2 and 3.3).
4. Remove the edges of the clique from the constraint graph.
5. Continue from step 1.
6. Represent the binary constraints corresponding to the remaining edges in the constraint graph explicitly as 2-literal clauses.

In the next sections we discuss three alternative representations of cliques of the constraint graph, starting from the trivial explicit representation.

3.1 Explicit $O(n^2)$ Representation

For a set C of literals forming a clique in the constraint graph the most obvious representation is as 2-literal clauses

$$\{l \vee l' | \{l, l'\} \subseteq C, l \neq l'\}.$$

This quadratic representation of the clique is illustrated in Figure 3 by 5 mutually exclusive literals of the form $v = i$. The number of non-equivalent clauses is $\frac{n(n-1)}{2}$ for n literals.

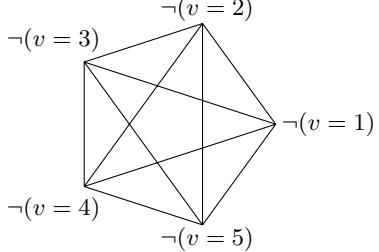


Figure 3. Representation of a clique of the constraint graph with $O(n^2)$ size and $O(1)$ additional propositional variables. An edge between literals $\neg(v = i)$ and $\neg(v = j)$ denotes their disjunction.

The $O(n^2)$ explicit representation of the constraints becomes less and less practical as n increases.

3.2 $O(n)$ Representation with $O(n)$ Auxiliary Variables

We have devised a linear-size representation that uses a linear number of auxiliary variables. This representation is illustrated in Figure 4. The representation can be viewed as a variant of the linear-size representations of interference constraints presented by Rintanen et al. [14] and the representation of many-valued propositions by Giunchiglia et al. [7].

The idea is to impose an arbitrary ordering on the n literals, and starting from each literal introduce a sequence of implications to the right through a sequence of auxiliary variables, leading to the negation of each literal on the right. When one of the literals is true, all of the auxiliary variables right from it must be true, and consequently all of the literals to the right must be false. This guarantees that if one of the literals is set true, all others are set false simply by applications of the unit resolution rule employed by most satisfiability algorithms.

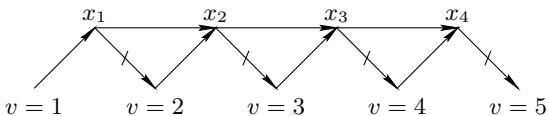


Figure 4. Representation of a clique in terms of $O(n)$ clauses and additional variables. An edge \rightarrow from l to l' denotes the formula $l \rightarrow l'$ and an edge $\not\rightarrow$ from l to l' denotes the formula $l \rightarrow \bar{l}'$.

The number of clauses in this linear-size representation is $3n - 4$ for $n \geq 2$. For 5 values the quadratic-size and linear-size representations respectively require 10 and 11 2-literal clauses, for 6 values respectively 15 and 14 clauses. Hence starting from 6 values the linear-size representation is smaller than the quadratic-size representation.

3.3 $O(n \log n)$ Representation with $O(\log n)$ Auxiliary Variables

Although the previous representation has an asymptotically optimal $O(n)$ size, the linear number of additional auxiliary variables may be considered high. Indeed, we can substantially decrease the number of auxiliary variables with the cost of a small increase in size.

The next representation has $\mathcal{O}(n \log n)$ size and $\mathcal{O}(\log n)$ auxiliary variables. Let $C = \{l_0, \dots, l_{n-1}\}$ be a clique consisting of n literals. Let x_0, \dots, x_{m-1} be new $m = \lceil \log_2 n \rceil$ Boolean variables. The constraint that only one literal can be true at a time is imposed by associating a different binary number with each literal. Since these binary numbers are represented in terms of the same variables, at most one of the literals can be true at a time. The representation is

$$\begin{aligned} l_0 &\rightarrow (\text{bit}_0(0) \wedge \text{bit}_1(0) \wedge \dots \wedge \text{bit}_{m-1}(0)) \\ l_1 &\rightarrow (\text{bit}_0(1) \wedge \text{bit}_1(1) \wedge \dots \wedge \text{bit}_{m-1}(1)) \\ &\vdots \\ l_{n-1} &\rightarrow (\text{bit}_0(n-1) \wedge \text{bit}_1(n-1) \wedge \dots \wedge \text{bit}_{m-1}(n-1)) \end{aligned}$$

where $\text{bit}_i(j) = x_i$ if the i th bit of j is 1 and $\text{bit}_i(j) = \neg x_i$ if the i th bit of j is 0.

For n literals the number of the resulting 2-literal clauses is $n \lceil \log_2 n \rceil$. For 5 values the numbers of clauses are respectively 15 and 11 for this representation and the linear-size representation, and numbers of auxiliary variables are respectively 3 and 4. For 32 values the numbers of clauses are respectively 160 and 92 and number of auxiliary variables are respectively 5 and 31.

This representation has been used by Frisch and Peugniez [5] and in more recent works [1], and the idea of two redundant representations connected by channelling constraints is familiar from the CSP context [4, 15].

4 Bicliques of the Constraint Graph

The presence of cliques in constraint graphs seems to be tied to particular classes of planning, model-checking and constraint satisfaction problems, and cannot be claimed to be extremely common. Indeed, in standard planning benchmarks, cliques of size n restrict an n -valued state variable, which is represented in terms of n Boolean state variables, to have at most one value at a time.

There is a need for techniques for compactly representing more general classes of constraint graphs. It is easy to see by a simple combinatorial argument that most constraint graphs do not have a compact representation, but there are many practically important graphs that do have many regularities that may be taken advantage of.

It seems that bicliques in constraint graphs are much more common than cliques. First, by partitioning any clique C to two disjoint sets C_1 and C_2 yields a biclique C_1, C_2 . So there are bicliques whenever there are cliques. Second, consider a subgraph C that is almost a clique but not quite. Let E' be the missing edges so that $\{\{n, n'\} | n \in C, n' \in C, n \neq n'\} \subseteq E \cup E'$. Now $C' = C \setminus \bigcup_{e \in E'} e$ is a clique, $C \setminus C'$, C' is a biclique because all the missing edges are between nodes in $C \setminus C'$, and $C \setminus C'$ may contain non-trivial bicliques.

A powerful discovery is that bicliques can be represented very compactly. When the literal sets C and C' form a biclique, the $|C| \cdot |C'|$ binary constraints can be represented by only $|C| + |C'|$ 2-literal clauses and only one auxiliary variable. For big bicliques this is a very big size reduction. This is illustrated in Figure 5. Notice that the original clause set can be obtained by resolving every pair of clauses that contain x and $\neg x$.

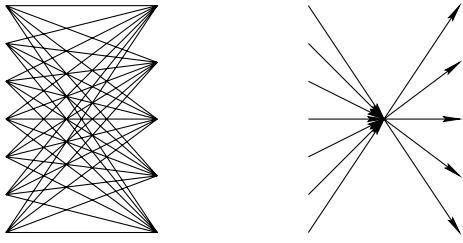


Figure 5. Representation of nm binary constraints forming an n, m biclique (left) in terms of only $n + m$ 2-literal clauses and one auxiliary variable (right). On the left, an edge between l and l' denotes the formula $l \vee l'$ (equivalently $\bar{l} \rightarrow l'$). On the right, an edge \rightarrow from l to the auxiliary variable x denotes $\bar{l} \rightarrow x$ and an edge from x to l' denotes $x \rightarrow l'$.

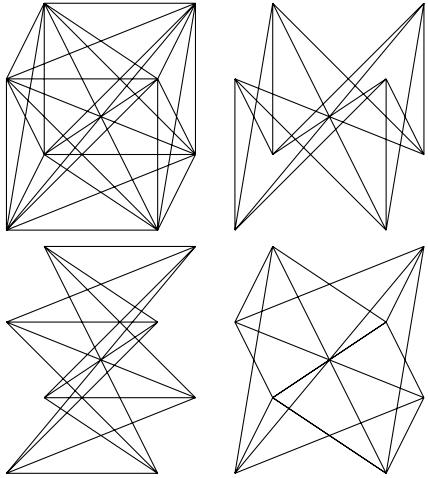


Figure 6. A clique of 8 nodes and 28 edges (upper left) decomposed to three 4,4 bicliques with 16 edges each. Division is along the up-down vertical axis (upper right), along the left-right horizontal axis (lower left), and along the front-back horizontal axis (lower right).

4.1 Relation to Compact Clique Representation

Decomposition of constraint graphs to cliques and their compact representation with only $O(n \log n)$ binary clauses can be understood in terms of bicliques. Hence the clique representation is subsumed by the biclique representation. A clique of n nodes corresponds to $\lceil \log_2 n \rceil$ orthogonal bicliques. Figures 6 and 7 illustrate the reduction from 28 edges of the original clique ($O(n^2)$) to 24 edges ($O(n \log n)$) in the biclique representation. This can also be illustrated by viewing the 8 nodes as 3-bit binary numbers. Now the three bicliques correspond to partitioning the 8 numbers by the values of the three digits. The partitions and bicliques are according to the first bit $\{000, 001, 010, 011\}$, $\{100, 101, 110, 111\}$, the second bit $\{000, 001, 100, 101\}$, $\{010, 011, 110, 111\}$ and the third $\{000, 010, 100, 110\}$, $\{001, 011, 101, 111\}$. The compact representation of these bicliques exactly corresponds to the $n \log n$ representation of cliques from Section 3.3.

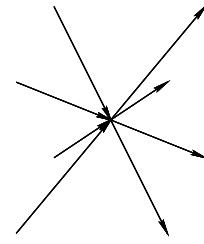


Figure 7. The invariants of the lower left biclique in Figure 6 represented more compactly in terms of an intermediate node.

5 Application: AI Planning

In most of the standard planning benchmarks all of the edges in the constraint graph stem from many-valued state variables that are encoded in terms of binary variables: the graph consists of cliques which say that each many-valued state variable can have only one value at a time. The representations from Section 3 encode these graphs compactly.

For the blocks world problems, if there are n blocks, the constraint that there can be at most one block below and above a block requires $2 \times \frac{n(n-1)}{2} = n(n-1)$ 2-literal clauses. For representing these cliques compactly by using the $O(n)$ representation only $6n - 8$ 2-literal clauses are needed. For 20 blocks this means a reduction from $20 \times 20 \times 19 = 7600$ to $20 \times (6 \times 20 - 8) = 2240$ clauses. For 40 blocks the reduction is from 62400 to 9280.

However, for many other problems the structure of constraint graphs is more complex and there are few cliques. One problem with only few big cliques is the airport scheduling problem from the 2004 planning competition. Most of this problem series can be efficiently solved by the planning as satisfiability approach but the biggest instances require the generation of huge propositional formulae with sizes of several gigabytes that exceed the physical memory restrictions of current 32-bit microprocessors. The invariants constitute more than 90 per cent of all the clauses in the CNF representation of the problems, and only a fraction of them forms cliques. The locations of airplanes correspond to many-valued state variables, but invariants on a number of related state variables that control the legal movement of airplanes through the airport do not.

We have very successfully used bicliques for making the representation of the invariants more compact. See Table 1. The size reduction of the formulae by a factor of about 9 makes it possible to generate all the formulae required for solving the problem series, and solve almost all of them efficiently save some of the last ones.

Interestingly, despite the big reduction in the formula sizes, those instances that were solvable before the reduction in formula size, the SAT solver (HaifaSAT, Siege) runtimes are improved not at all or only minimally. The biggest reductions in total runtimes are due to faster disk I/O and processing times during formula generation.

In addition to the airport problems, for other problems substantial reductions in the number of clauses for invariants are obtained, but the total reductions in the formula sizes are smaller, for example for the 30 block blocks world problems 50 per cent, because the invariants do not as strongly dominate the size of the formulae. For the simplest problems our biclique algorithm finds all maximal bicliques. This is probably not true for more complicated problems, for example the airport problem. Hence it would be interesting to test algorithms with approximation guarantees [8].

instance	clauses for invariants		size in MB	
	before	after	before	after
21_4halfMUC_P2	182094	13191	2.59	0.37
22_4halfMUC_P3	275927	21388	4.06	0.58
23_4halfMUC_P4	381675	31776	5.60	0.84
24_4halfMUC_P4	383791	30407	5.72	0.90
25_4halfMUC_P5	478455	41719	7.24	1.18
26_4halfMUC_P6	587951	50247	8.85	1.43
27_4halfMUC_P6	572292	53721	9.01	1.57
28_4halfMUC_P7	670530	66060	10.62	1.89
36_5MUC_P2	325136	18872	4.68	0.52
37_5MUC_P3	490971	30681	7.40	0.93
38_5MUC_P3	487600	29464	7.30	0.86
39_5MUC_P4	655616	44647	10.08	1.34
40_5MUC_P4	657309	43872	10.04	1.27
41_5MUC_P4	653940	42314	9.93	1.20

Table 1. Size reduction by the biclique representation. For each problem instance we give the number of clauses when each invariant is represented by one 2-literal clause and when bicliques of the constraint graph are represented compactly. The formula sizes are for DIMACS-encoded CNF formulae per time point, including the invariants and the rest of encoding of the planning problem. For example for 100 time points the total formula size is obtained by multiplying by 100. The first problem instance has 1696 state variables, the last has 4957. Solving the last instances of the series (42 to 50) require formulae with sizes up to 8 gigabytes which do not fit in the memory of a computer with a 32-bit address space. The shortest plan for instance 22 has length 53 and the corresponding formula without compression is 222 megabytes in the DIMACS CNF format, with clique compression 187 megabytes, and with biclique compression 31 megabytes. The numbers of clauses for invariants are respectively 275927, 231673 and 21388.

We also made a small experiment with constraint graphs that represent interference constraints for preventing simultaneous actions if they interfere [11]. These constraint graphs have in the worst case n^2 edges for n actions but there is a specialized representation with an asymptotically optimal linear size [14]. The biclique representation was often moderately or much smaller than the explicit and the specialized $\mathcal{O}(n)$ representation, but in some cases there was very little reduction and the specialized $\mathcal{O}(n)$ representation was by far the most compact. This possibility makes the biclique representation unattractive for this application.

6 Conclusions and Related Work

We have considered the problem of representing big constraint graphs more compactly based on representations that involve decomposing parts of the graphs to cliques and bicliques. The biclique representation, which is the new contribution of this work, turned out to be the more general representation as the clique representation is subsumed by it.

Constraint graphs with n nodes may have n^2 edges and our compact biclique representation does not improve this $\mathcal{O}(n^2)$ worst-case size. However, our experiments in Section 5 show that for practically interesting constraint graphs big size reductions are possible, by a factor of 10, and this may be the difference between an impractically big SAT problem (with respect to the memory requirements) and a relatively easily solvable one.

Acknowledgements

This research was supported by DFG grant RI 1177/2-1 (during the work at the Albert-Ludwigs-Universität Freiburg) and by National ICT Australia (NICTA). NICTA is funded through the Australian

Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

REFERENCES

- [1] Carlos Ansótegui and Felip Manya, ‘Mapping problems with finite-domain variables into problems with Boolean variables’, in *SAT 2004 - Theory and Applications of Satisfiability Testing: 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, eds., Holger Hoos and David G. Mitchell, number 3542 in Lecture Notes in Computer Science, pp. 1–15. Springer-Verlag, (2005).
- [2] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu, ‘Symbolic model checking without BDDs’, in *Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of 5th International Conference, TACAS'99*, ed., W. R. Cleaveland, volume 1579 of *Lecture Notes in Computer Science*, pp. 193–207. Springer-Verlag, (1999).
- [3] Avrim L. Blum and Merrick L. Furst, ‘Fast planning through planning graph analysis’, *Artificial Intelligence*, **90**(1-2), 281–300, (1997).
- [4] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu, ‘Increasing constraint propagation by redundant modeling: an experience report’, *Constraints*, **4**, 167–192, (1999).
- [5] Alan M. Frisch and Timothy J. Peugniez, ‘Solving non-Boolean satisfiability problems with stochastic local search’, in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, ed., Bernhard Nebel, pp. 282–288. Morgan Kaufmann Publishers, (2001).
- [6] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, 1979.
- [7] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner, ‘Nonmonotonic causal theories’, *Artificial Intelligence*, **49**–104, (2004).
- [8] Dorit S. Hochbaum, ‘Approximating clique and biclique problems’, *Journal of Algorithms*, **29**, 174–200, (1998).
- [9] R.M. Karp, ‘Reducibility among combinatorial problems’, in *Complexity of Computer Computations*, eds., R.E. Miller and J. W. Thatcher, pp. 85–103. Plenum Press, (1972).
- [10] Henry Kautz and Bart Selman, ‘Planning as satisfiability’, in *Proceedings of the 10th European Conference on Artificial Intelligence*, ed., Bernd Neumann, pp. 359–363. John Wiley & Sons, (1992).
- [11] Henry Kautz and Bart Selman, ‘Pushing the envelope: planning, propositional logic, and stochastic search’, in *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pp. 1194–1201. AAAI Press, (August 1996).
- [12] René Peeters, ‘The maximum edge biclique problem is NP-complete’, *Discrete Applied Mathematics*, **131**(3), 651–654, (2003).
- [13] Jussi Rintanen, ‘A planning algorithm not based on directional search’, in *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, eds., A. G. Cohn, L. K. Schubert, and S. C. Shapiro, pp. 617–624. Morgan Kaufmann Publishers, (June 1998).
- [14] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä, ‘Planning as satisfiability: parallel plans and algorithms for plan search’, Report 216, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, (2005).
- [15] Barbara. M. Smith, K. Stergiou, and T. Walsh, ‘Using auxiliary variables and implied constraints to model non-binary problems’, in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000) and the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, pp. 182–187. AAAI Press, (2000).
- [16] M. Yannakakis, ‘Node- and edge-deletion NP-complete problems’, in *Proceedings of the Tenth Annual ACM Symposium on the Theory of Computing*, pp. 253–264. The Association for Computing Machinery, (1978).

Pessimistic Heuristics Beat Optimistic Ones in Real-Time Search

Aleksander Sadikov and Ivan Bratko¹

Abstract. Admissibility is a desired property of heuristic evaluation functions, because when these heuristics are used with complete search methods, such as A* and RBFS, they guarantee that an optimal solution will be found. Since every optimistic heuristic function is admissible, optimistic functions are widely used. We show, however, that with incomplete, real-time search, optimistic functions lose their appeal, and in fact they may hinder the search under quite reasonable conditions. Under these conditions the exact opposite is to be preferred, i.e. pessimistic heuristic functions that never *underestimate* the difficulty of the problem. We demonstrate that such heuristics behave better than optimistic ones of equal quality on a standard testbed using RTA* search method.

1 INTRODUCTION

Admissibility is a desired property of heuristic evaluation functions, because when these heuristics are used with complete search methods, such as A* [6] and RBFS [8], they guarantee that an optimal solution will be found. A known theorem about the admissibility states that a heuristic function, which always gives an optimistic assessment of the position, is admissible [6]. This theorem made optimistic heuristic functions popular and widely used. The main problems with complete search methods, though, are their exponential running time and the necessity to wait until the search completes before the first step towards the goal can be taken. To alleviate these problems, incomplete search methods have been proposed, such as RTA* [7]. Since these methods can tackle much larger problems, they can often be the only practical option.

Incomplete search methods do not guarantee finding an optimal solution even when used in conjunction with admissible heuristics. Thus, the main reason for using admissible and consequently optimistic heuristics is void. Nevertheless, people use optimistic heuristic functions with incomplete search methods, because: (a) they are often readily available, since they were developed for complete search algorithms, (b) common sense saying that since they proved useful with complete search methods, perhaps they are useful with incomplete search methods as well, and (c) it was never shown that they could be counterproductive.

In this paper we show that under very reasonable conditions, optimistic functions are counterproductive in real-time (incomplete) search. Our results indicate that pessimistic functions should be used instead.

Section 2 of the paper derives the condition under which optimistic heuristics harm real-time search. It also shows that pessimistic

heuristics behave much better under these same conditions. An intuitive explanation for this phenomenon is given. Section 3 compares pairs of heuristic functions of equal quality, one optimistic, the other pessimistic, and demonstrates that the latter give much better results when used with RTA* search in the 8-puzzle domain. Section 4 discusses some consequences of this finding. Section 5 concludes the paper and gives some pointers for further work.

2 OPTIMISTIC AND PESSIMISTIC HEURISTICS

Let RTA* evaluate node n with an evaluation function f of the common form $f(n) = g(n) + h(n)$. Here $g(n)$ is the cost of the path from the current node to node n , and $h(n)$ is a heuristic estimate of the cost of an optimal path from node n to the nearest goal node. Let us denote the true cost of such an optimal path from node n with $h^*(n)$. The relationship between the true and heuristic value of node n is governed by the following equation:

$$h(n) = h^*(n) + e(n) \quad (1)$$

where $e(n)$ is an error the heuristic makes at node n .

Suppose the current state has two successors, nodes a and b , a being better than b . The immediate task of the search is to choose between node a and node b . RTA* will choose a if $f(a) < f(b)$. We will simplify the analysis slightly by assuming without loss that all the edges have unit cost. Then the condition $f(a) < f(b)$ for choosing a is equivalent to the condition $h(a) < h(b)$. If this condition holds then RTA* will make the correct decision (i.e. choose a), otherwise it will commit a decision error by choosing b . So a decision error occurs when

$$h(a) > h(b). \quad (2)$$

This can be rewritten in terms of true costs h^* and heuristic errors e :

$$h^*(a) + e(a) > h^*(b) + e(b). \quad (3)$$

Since a is better than b , $h^*(a) < h^*(b)$, and the difference $\Delta h^* = h^*(b) - h^*(a)$ is positive. The condition for decision error can be stated as:

$$e(a) - e(b) > \Delta h^* > 0. \quad (4)$$

Now let us consider what happens in two special cases: (1) when the algorithm uses an *optimistic* heuristic function, and (2) when it uses a *pessimistic* heuristic function. By definition, for optimistic heuristics, heuristic errors e are always negative because heuristic approximations underestimate true costs. So for optimistic heuristics, for all nodes n , $e(n) = -|e(n)|$. The decision error condition then becomes:

$$|e(b)| - |e(a)| > \Delta h^* > 0. \quad (5)$$

¹ University of Ljubljana, Faculty of Computer and Information Science, Artificial Intelligence Laboratory, Tržaška 25, 1000 Ljubljana, Slovenia, e-mail: {aleksander.sadikov;ivan.bratko}@fri.uni-lj.si

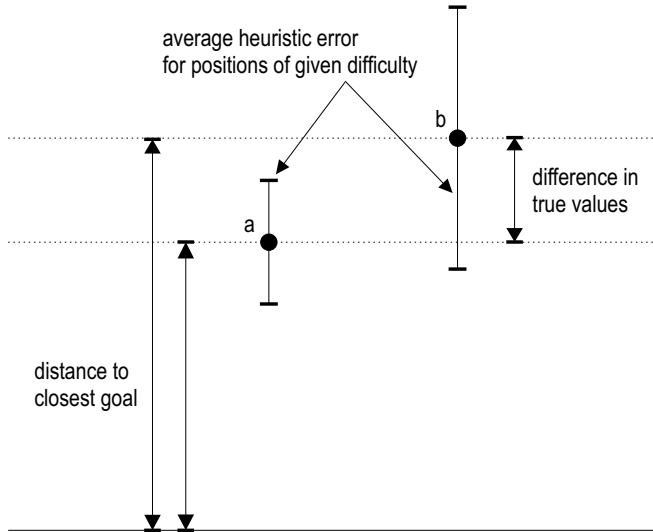


Figure 1. The difference in true values

Equation 5 has an interesting consequence. It basically says that if the heuristic function's absolute error increases with increasing difficulty of the problems (nodes requiring more costly paths to reach the goal), then for optimistic heuristics the chance to make a decision error *increases*. In our example, node b is more difficult than a , so under this assumption, absolute heuristic error at b will tend to be greater than at a . This indicates a problem with optimistic heuristics when heuristic errors (in absolute terms) increase with the difficulty of the problems. Conversely, if heuristic errors decrease with the difficulty of the problems, then the optimistic heuristic will have better chance of making correct decisions. However, as also discussed later in the paper, this second case (heuristic error decreasing with the cost) seems to be rather unrealistic in practice.

For pessimistic heuristics, by definition heuristic errors e are always positive, and the condition for RTA* making a decision error is:

$$|e(a)| - |e(b)| > \Delta h^* > 0. \quad (6)$$

If we compare Equations 5 and 6, we can see that just the opposite of what is true for optimistic heuristics holds for pessimistic heuristics. Pessimistic heuristics have better chances to produce correct decision if the heuristic error increases with increasing difficulty of the nodes, and worse if the heuristic error decreases with increasing difficulty of the nodes. According to this, in the case of absolute error increasing with difficulty, i.e. the case to be expected more likely in practice, optimistic heuristics will tend to make decision errors more frequently than pessimistic heuristics.

2.1 Intuitive explanation

Figure 1 shows a decision task we discussed. The tails stemming from nodes a and b represent the average heuristic error for the problems of given difficulty. To commit a decision error, the heuristic values have to overcome the difference in true values between the two nodes. Suppose the heuristic's errors increase with increasing difficulty of the problems. Since node b represents a more difficult

problem, the heuristic will on the average make a larger error in it. This is represented in the figure with a longer tail coming out of node b than the one coming out of node a . Furthermore, if the heuristic is optimistic, only downward part of the tail is possible, and vice versa, if the heuristic is pessimistic only the upward part of the tail is possible. If the heuristic is optimistic, the longer tail, coming out of node b has to overcome the difference in true values to give a chance for a decision error to be committed. However, if the heuristic is pessimistic, it is the shorter tail coming out of node a that has to overcome the difference in true values. Since the longer tail can more easily overcome this difference than the shorter tail, it follows that the optimistic heuristic can more easily commit a decision error when the heuristic's errors increase with increasing difficulty of the problems.

3 EXPERIMENTS

We have tested our theoretical results on a classical testbed for single-agent search methods, the 8-puzzle sliding tiles problem, described e.g. in [2]. We have chosen this small puzzle, because for complete evaluation of success of search we needed to know the true value of every state in the domain — the reason for this will become apparent in the next subsection.

3.1 The two heuristic functions

To empirically test our theoretical findings, we compared two heuristic functions, one optimistic, the other pessimistic. The functions needed to be of equal quality (in terms of their relative errors), so that neither one of them has an unfair advantage. To get such a pair of functions, we had to artificially construct them.

We first calculated the true values of all legal 8-puzzle positions with the use of retrograde analysis, a technique known from computer chess, where it is used to generate endgame databases [13]. An indexed array of distances to nearest goal defines the perfect evaluation function h^* . Then we proceeded to generate the two heuristic functions by appropriately corrupting our perfect heuristics.

We modelled the distribution of our optimistic heuristic after the distribution of Manhattan distance heuristic — a well-known optimistic heuristic for the 8-puzzle domain. On average, Manhattan distance heuristic's error increases with increasing difficulty of the problems (the average error over all positions of a given difficulty level). We measure the difficulty of the problem as the number of steps needed to reach the goal assuming optimal play, i.e. with the problem's true value. The dispersion of heuristic values around the average evaluation for a given difficulty level is more or less constant.

We created our artificial heuristics by corrupting the perfect evaluations in a controlled manner. Our method of doing this is as follows. We take a position's true value $h^*(n)$ and add to it a certain amount of Gaussian noise, described with the formula:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}. \quad (7)$$

The formula gives the probability $P(x)dx$ that given the correct evaluation $\mu = h^*(n)$ and standard deviation σ , the heuristic evaluation, $h(n) = x \in \mathbf{R}$, will take on a value in the range $[x, x + dx]$. The error of heuristic evaluation $h(n)$ is $e(n) = x - \mu$. We do this for all legal positions. A more detailed description of this process is given in [11]. Parameter σ controls the level of corruption. Since we modelled our heuristics after the Manhattan heuristic, we chose $\sigma = 2.5$ steps to equal the standard deviation of Manhattan heuristic's evaluations.

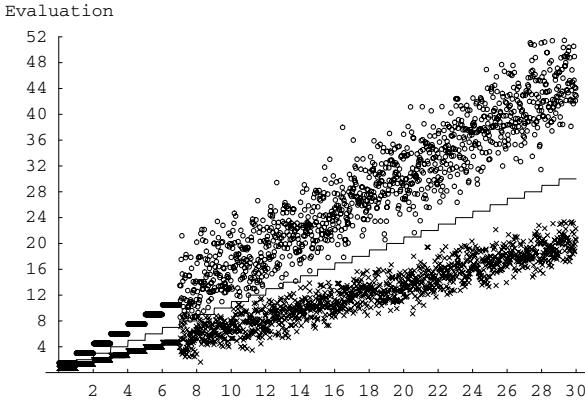


Figure 2. The evaluations given by our heuristic functions

To get the optimistic heuristic, we multiplied the obtained corrupted heuristic by a constant c :

$$h(n) = c \cdot (h^*(n) + e(n)). \quad (8)$$

We chose $c = 2/3$ to emulate the level of errors Manhattan heuristic commits. The pessimistic heuristic of comparable relative error was obtained by multiplying the corrupted heuristic by inverted constant $1/c$, that is with $3/2$. The random process of corrupting the true evaluations was of course repeated for the pessimistic heuristic (we did not use the same errors as with the optimistic heuristic). If some evaluations in either heuristic were not pessimistic or optimistic as intended, their values were corrupted again.

Both heuristics are plotted in Figure 2. The x-axis gives the difficulty (true value) of the position, the y-axis gives the heuristic evaluation of the position. The crosses represent the optimistic heuristic, the circles the pessimistic heuristic, and the solid line represents the true evaluations. A random sample of 50 positions of each difficulty is displayed. The figure clearly shows that the average heuristic error grows close to linear with increasing difficulty of positions for both heuristics.

The first seven levels of difficulty deserve an explanation. For these levels we did not corrupt the true evaluations, we just multiplied them with the appropriate constant. The reason for this is that few positions belong to these levels and it is therefore practically impossible to corrupt them so that they would maintain more or less constant dispersion. Thus, we once again decided to model after the Manhattan distance heuristic, which also without exception gives correct estimates for the first seven levels of difficulty.

3.2 The search engine

We varied the depth of RTA* lookahead from 1 to 30, thirty being the difficulty of the hardest 8-puzzle problems. We were able to reach these very high depths of lookahead by employing transposition tables — yet another technique known from computer chess. We took advantage of the comparatively small number of possible positions in the 8-puzzle domain. We calculated depth 1 lookahead evaluations for all positions and stored these values in an array. Then we calculated depth 2 evaluations for all positions by doing one ply of search and using previously stored depth 1 evaluations, again storing

the calculated evaluations. We repeated the process for other depths. A more detailed description of the procedure is given in [11].

3.3 Results

We were interested in two characteristics: the percentage of correct decisions each of our heuristics makes when used in conjunction with RTA* search and the actual solution length such a search produces.

3.3.1 Percentage of correct decisions

When measuring the percentage of correct decisions we varied the difficulty of the problems to be solved and the depth of lookahead. For a given difficulty level of problems and a given depth of lookahead, we measured the average percentage of correct decisions on *all* possible puzzles of this level where one path is clearly better from the others (otherwise there is nothing to decide between).

The results of the experiments are presented in Figure 3. The x-axis represents the depth of lookahead, and the y-axis represents the percentage of correct decisions made. Figure 3 shows a representative subset of the results. The chart on the left represents moderately difficult puzzles, the middle one hard puzzles, and the right chart represents a random mixture of 1,000 puzzles of various difficulties — this way of testing is quite common and was used for example in [3, 7]. The first two charts do not represent a single puzzle, but rather all puzzles of the given difficulty.

It is obvious from the charts that the pessimistic heuristic, represented by a dashed line, gives rise to a higher average percentage of correct decisions than the optimistic heuristic for all difficulty levels of the puzzles. This is especially so when the lookahead depth is close to the difficulty level of puzzles.

One may argue that perhaps the constant $c = 3/2$ used to get the pessimistic heuristic is misguided, and that $c = 4/3$ should be used instead. The latter gives the same level of errors in absolute terms, while the first one gives the same relative errors. We indeed constructed such a pessimistic heuristic as well, and the results were, as expected, even more in favour of the pessimistic heuristic.

3.3.2 Solution length

When measuring the solution length we varied the depth of lookahead and the quality of heuristics used (by varying the parameter σ for the pair of heuristic functions). The results are presented in Figure 4. The x-axis again represents the depth of lookahead, and the y-axis represents the solution length. The results are averaged over all legal puzzles. The dotted line at 21.50 represents the length of optimum solution averaged over all legal puzzles. The left chart represents the case for $\sigma = 2.5$ (the pair of heuristics modelled after the Manhattan heuristic), while the other two charts represent the cases with larger heuristic errors, $\sigma = 3$ and $\sigma = 4$, respectively.

We can see that pessimistic heuristics (dashed line) clearly outperform their optimistic counterparts (solid line) by consistently finding shorter solutions. It is interesting that after the depth of lookahead reaches 5 moves, the gain in solution length (thick line) is quite constant. At very high search depths the gain of course decreases, eventually reaching zero, because more and more puzzles are solved optimally since the solution is seen directly from the starting state.

The gain in solution length for the pessimistic heuristic over its optimistic counterpart, modelled by Manhattan heuristic, is about 5% to 10% (on the interesting interval with lookahead depth over 5 and before too many solutions are directly found). The decreasing quality

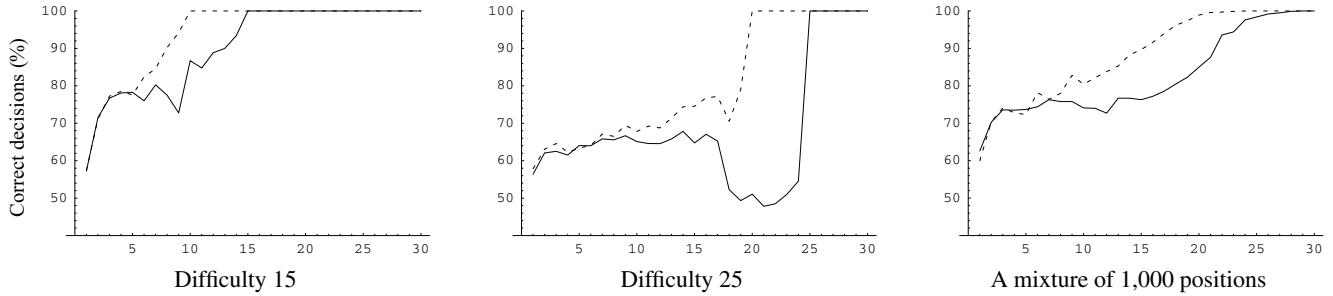


Figure 3. The comparison of correct decisions made between optimistic (solid line) and pessimistic (dashed line) heuristic

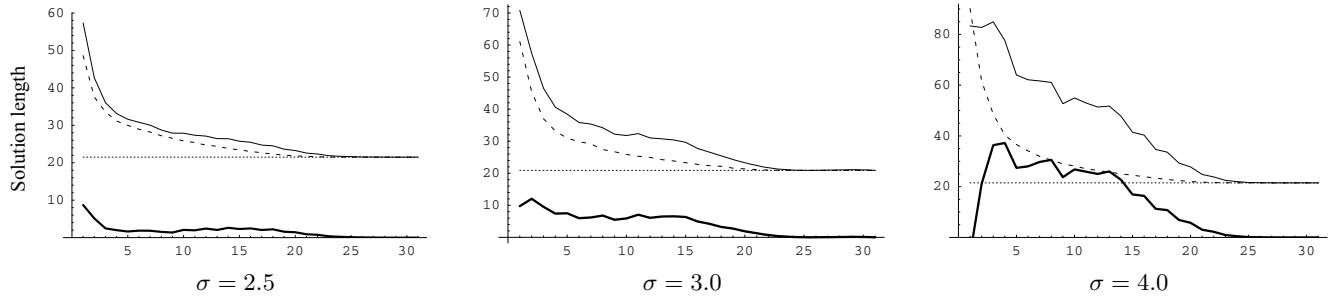


Figure 4. The comparison of solution length between optimistic (solid line) and pessimistic (dashed line) heuristic; the thick line represents the difference between them (the gain by pessimistic heuristic)

of heuristics, however, causes a sharp increase in the gain. For $\sigma = 3$ the gain is slightly below 20% and for $\sigma = 4$ it is already about 50%.

3.4 Search depth pathology

A search depth pathology is a phenomenon when deeper lookahead results in more decision errors committed or worse solutions found. This pathology was first discovered in two-player games in the late 1970s [1, 9]. An overview is given in [10, 11]. Recently, such pathology has also been detected in single-agent search [3, 5]. As we can see from the charts in Figures 3 and 4 our artificial heuristics also display such pathological behaviour. However, it is interesting that the pessimistic heuristic displays lesser inclination towards such behaviour than its optimistic counterpart. As we can see the pathological behaviour is not limited only to decision quality but also manifests itself in solution length. This was also observed by Korf [7]. With lower quality of heuristics the optimistic ones become more and more pathological while the pessimistic ones behave normally. This deserves further study.

4 DISCUSSION

We have shown that if the error committed by the heuristic evaluation increases with the difficulty of the problems, pessimistic heuristics are comparatively more successful than optimistic ones of equal quality, and vice versa, if the heuristic error decreases with the difficulty of the problems, then optimistic heuristics are better. This gives

rise to an important question: which is more plausible, increasing or decreasing heuristic errors? We believe that the majority of real-life heuristics belong to the first group, namely that their heuristic errors increase with increasing difficulty of the problems. For example, imagine a task scheduling problem and a heuristic for it. If the optimal solution is that the tasks complete in, say, 10 hours, it is easy to imagine a heuristic estimating this time as somewhere between 8 and 12 hours, that is committing an error of ± 2 hours. However, if the optimal solution is that the tasks complete in 1,000 hours, it is quite unimaginable that the heuristic would estimate this time as somewhere between 998 and 1,002 hours. That would be an incredibly accurate heuristic. It is much more likely that the heuristic would estimate the time needed as something between 800 and 1,200 hours, committing the error of ± 200 hours. We believe it is quite conceivable that the error is usually *relative* to the size/difficulty of the problem; that the heuristic is, say, 20% off the mark. However, the equations in Section 2 only need this error to increase in absolute terms, it does not matter that the heuristic is more or less equally wrong in relative terms. From this we conclude that pessimistic heuristics seem to be preferable. We can give one more example: to which group do the two well-known heuristics for the 8-puzzle belong? Both, Manhattan distance and “Manhattan distance + Sequence score” heuristics, described e.g. in [2], belong to the first group — their error rises with increasing difficulty of the problems.

A possible argument in favour of optimistic heuristics could be that they may be easier to construct than pessimistic ones. But is this really so? Probably it is just that people are more used to opti-

mistic heuristics since they were usually preferred. For example, it is trivial to use air distance as a heuristic when approximating the road distance between two cities. But it is similarly trivial to use as a heuristic the distance based solely on highways, not on all the roads. This heuristic is pessimistic. Another point should be noted here. The equations in Section 2 do not say that it is strictly necessary for a heuristic to be pessimistic for *every* position, just for most of them. This, on the other hand, is not the case with the admissibility theorem — it necessitates that *all* positions are optimistically evaluated for the heuristic to be admissible.

The difference in the decision accuracy between the two heuristic functions in our experiments is a few percent, unless the search reaches the vicinity of goal nodes, where this difference increases. How much are these few percent worth? Not so little, because the search makes a series of decisions and for every single one of them the pessimistic heuristic gives it an extra few percent. For example, in the 8-puzzle domain, each mistake means the search will have to make at least two additional steps — one going back to the position where it came from, the other taking the right path (which it could have taken in the first place if it would not make a mistake).

Suppose we have an optimistic heuristic for some problem, e.g., the Manhattan distance for the 8-puzzle. We could make it pessimistic by adding a constant to it. Unfortunately this approach does not work, because by doing so the new heuristic's error would actually increase with decreasing problem difficulty, so all we would do is transform one disadvantageous case into another. How about multiplying an optimistic heuristic by a constant to get a pessimistic one?² This approach might actually work, though not for every heuristic. In some cases, like with Manhattan distance, we could again get a pessimistic heuristic whose error again increases with decreasing difficulty of the problems. But for some heuristics we could succeed. But even in this case we would have to experiment with finding the right constant, since if the constant is too big, we degrade the heuristic too much, and if the constant is too small, the resulting heuristic might not be pessimistic.

In section 2, we showed that RTA* using optimistic heuristics is likely to make more decision errors than RTA* with pessimistic heuristics. A related question is how the fixed-depth lookahead performed by RTA* affects the accuracy of heuristic evaluations. When RTA* decides which node to move to next, it considers the minimin backed-up heuristic values rather than the original, “static” heuristic values. The relevant question is: Are these backed-up heuristic values more accurate than the static heuristic estimates themselves? We will here indicate, without full mathematical proof, the intuition behind the following answer: Minimin backed-up heuristic values have on average smaller absolute error when the terminal nodes of lookahead are evaluated by pessimistic heuristics than when they are evaluated optimistically. Of course in all detail this depends on the distribution of true values and on error distribution. It is however easy to see the advantage of pessimistic heuristics in the cases when there are several lookahead terminal nodes all of which having true values close to the lowest true value. Minimin lookahead will return the minimum heuristic value encountered at the fringe of the lookahead. In the case of pessimistic heuristics, this minimum heuristic value will tend to be the *least distorted* among the true values close to optimal. This is because minimin will tend to return the least overestimated node close to optimal. On the contrary, in the case of optimistic heuristics, this

minimum heuristic value will tend to be the *most distorted* among the values close to optimal, because minimin will tend to return the most underestimated node value close to optimal.

5 CONCLUSIONS AND FURTHER WORK

We have shown that pessimistic heuristic functions are more effective than their optimistic counterparts of equal quality when used with incomplete search methods under the condition that the heuristic's errors grow with increasing difficulty of the problems. We have argued that such a condition is often met in practice, and that therefore pessimistic heuristics should be preferred.

We have also mentioned that our preconditions do not strictly necessitate that the heuristic is pessimistic for every single node. We believe it would be worthwhile to study how the percentage of such violations of pessimistic evaluation affect the gain.

The experiments also indicated that while both heuristic functions display some pathological behaviour, the pessimistic heuristics seem less inclined to do so. Why is this so is an interesting topic for further work.

ACKNOWLEDGEMENTS

This work was partly funded by ARRS, Slovenian Research Agency.

REFERENCES

- [1] Donald F. Beal, ‘An analysis of minimax’, in *Advances in Computer Chess 2*, ed., M. R. B. Clarke, pp. 103–109. Edinburgh University Press, Edinburgh, UK, (1980).
- [2] Ivan Bratko, *Prolog Programming for Artificial Intelligence*, Addison-Wesley Publishing Company, 3rd edn., 2001.
- [3] Vadim Bulitko, ‘Lookahead pathologies and meta-level control in real-time heuristic search’, in *Proceedings of the Fifteenth Euromicro Conference on Real-Time Systems*, pp. 13–16, Porto, Portugal, (July 2003).
- [4] Vadim Bulitko and Greg Lee, ‘Learning in real-time search: A unifying framework’, *Journal of Artificial Intelligence Research*, **25**, 119–157, (2006).
- [5] Vadim Bulitko, Lihong Li, Russ Greiner, and Ilya Levner, ‘Lookahead pathologies for single agent search’, in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, eds., G. Gottlob and T. Walsh, pp. 1531–1533, Acapulco, Mexico, (August 2003).
- [6] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael, ‘A formal basis for the heuristic determination of minimum cost paths’, *IEEE Transactions on Systems Science and Cybernetics*, **4**(2), 100–107, (1968).
- [7] Richard E. Korf, ‘Real-time heuristic search’, *Artificial Intelligence*, **42**(2-3), 189–211, (1990).
- [8] Richard E. Korf, ‘Linear-space best-first search’, *Artificial Intelligence*, **62**(1), 41–78, (1993).
- [9] Dana S. Nau, *Quality of Decision Versus Depth of Search on Game Trees*, Ph.D. dissertation, Duke University, Durham, NC, 1979.
- [10] Aleksander Sadikov, *Propagation of heuristic evaluation errors in game graphs*, Ph.D. dissertation, University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia, 2005.
- [11] Aleksander Sadikov, Ivan Bratko, and Igor Kononenko, ‘Bias and pathology in minimax search’, *Theoretical Computer Science*, **349**(2), 268–281, (2005).
- [12] Masashi Shimbo and Toru Ishida, ‘Controlling the learning process of real-time heuristic search’, *Artificial Intelligence*, **146**(1), 1–41, (2003).
- [13] Ken Thompson, ‘Retrograde analysis of certain endgames’, *ICCA Journal*, **9**(3), 131–139, (1986).

² Both, adding a constant or multiplying with a constant is used in the weighted LRTA* algorithm [4, 12]. However, while the purpose there is to allow *some* states to be evaluated pessimistically, here we would like *all* states to be evaluated pessimistically.

Inverse Consistencies for Non-binary Constraints

Kostas Stergiou¹ and Toby Walsh²

Abstract. We present a detailed study of two inverse consistencies for non-binary constraints: relational path inverse consistency (rel PIC) and pairwise inverse consistency (PWIC). These are stronger than generalized arc consistency (GAC), even though they also only prune domain values. We propose algorithms to achieve rel PIC and PWIC, that have a time complexity better than the previous generic algorithm for inverse consistencies. One of our algorithms for PWIC has a complexity comparable to that for GAC despite doing more pruning. Our experiments demonstrate that inverse consistencies can be more efficient than GAC on a range of non-binary problems.

1 INTRODUCTION

Local consistency techniques are of great importance in constraint programming. They prune values from the domain of variables and terminate branches of the search tree, saving much fruitless exploration of the search tree. Local consistency techniques that only filter domains like (generalized) arc consistency (GAC) tend to be more practical than those that alter the structure of the constraint graph or the constraints' relations (e.g. path consistency). For binary constraints, many domain filtering consistencies have been proposed and evaluated, including inverse and singleton consistencies [8, 6, 13]. The situation is very different for non-binary constraints. A number of consistencies that are stronger than GAC have been developed, including relational consistency [12] and pairwise consistency [9]. However, these are typically not domain filtering.

We study here two promising domain filtering techniques for non-binary constraints: relational path inverse consistency (rel PIC) and, the stronger, pairwise inverse consistency (PWIC). Relational consistencies [12] have rarely been used in practice as most are not domain filtering, and have high time complexities. Pairwise consistency [9] (also called inter-consistency [10]) has also rarely been used as it is not domain filtering and the algorithm proposed in [9] has a high time complexity and requires all constraints to be extensional. Rel PIC and PWIC are local consistency techniques that do not suffer from these problems. They are domain filtering and are not prohibitively expensive to enforce. Our theoretical analysis reveals some surprises. For example, when restricted to binary constraints, rel PIC does not reduce to the variable based definition of path inverse consistency.

We propose algorithms to enforce rel PIC and PWIC that can be applied to constraints intentionally or extensionally specified. Their time complexity is better than the complexity of the generic algorithm for inverse consistencies given in [13]. The time complexity of the second algorithm for PWIC is $O(e^2 k^2 d^k)$ where e is the number of constraints, k is the maximum arity and d is the maximum domain size. This is significantly lower than the complexity of the generic algorithm, and is comparable to the complexity

of standard GAC algorithms, like GAC-Schema ($O(ekd^k)$) [3] and GAC2001/3.1 ($O(ek^2 d^k)$) [5]. However, this improvement comes at a cost as the space required is exponential in the number of *shared* variables. Experimental results demonstrate that it is feasible to maintain rel PIC and PWIC during search, and they can be more cost-effective than GAC on certain problems.

2 BACKGROUND

A *Constraint Satisfaction Problem* (CSP) is defined as a tuple (X, D, C) where: X is a set of n variables, D is a set of domains, and C is a set of e constraints. Each constraint c_i is a pair $(var(c_i), rel(c_i))$, where $var(c_i) = \{x_{j_1}, \dots, x_{j_k}\}$ is an ordered subset of X , and $rel(c_i)$ is a subset of the *Cartesian* product $D(x_{j_1}) \times \dots \times D(x_{j_k})$. A tuple $\tau \in rel(c_i)$ is *valid* iff none of the values in the tuple has been removed from the domain of the corresponding variable. Any two constraints c_i and c_j *intersect* iff the set $var(c_i) \cap var(c_j)$ of variables involved in both constraints is not empty. We assume that the number of variables that any two constraints have in common (denoted by f) is uniform. This assumption is made to simplify the description and complexity analysis of the algorithms, and can be easily lifted.

The assignment of value a to variable x_i is denoted by (x_i, a) . The set of variables over which a tuple τ is defined is $var(\tau)$. For any subset var' of $var(\tau)$, $\tau[var']$ is the sub-tuple of τ that includes only assignments to the variables in var' . Any two tuples τ and τ' of $rel(c_i)$ can be lexicographically ordered. In this ordering, $\tau <_l \tau'$ iff there exists a subset $\{x_1, \dots, x_j\}$ of $var(c_i)$ such that $\tau[x_1, \dots, x_j] = \tau'[x_1, \dots, x_j]$ and $\tau[x_{j+1}] <_l \tau'[x_{j+1}]$. An assignment τ is *consistent* iff for all constraints c_i , where $var(c_i) \subseteq var(\tau)$, $\tau[var(c_i)] \in rel(c_i)$. A *solution* is a consistent assignment to all variables. A value $a \in D(x_j)$ is consistent with a constraint c_i , where $x_j \in var(c_i)$, iff $\exists \tau \in rel(c_i)$ such that $\tau[x_j] = a$ and τ is valid. In this case, we say that τ is a GAC-support of a in c_i . A constraint c_i is *Generalized Arc Consistent* (GAC) iff $\forall x_j \in var(c_i)$, $\forall a \in D(x_j)$, there exists a GAC-support for a in c_i . A problem is GAC iff domains are non-empty and all constraints are GAC.

A binary problem is (i, j) *consistent* iff it has non-empty domains and any consistent instantiation of i variables can be extended to a consistent instantiation involving j additional variables [7]. A problem is *strong* (i, j) -consistent iff it is (k, j) consistent for all $k \leq i$. A problem is *arc consistent* (AC) iff it is $(1, 1)$ -consistent. A problem is *(strong) path consistent* (PC) iff it is *(strong) $(2, 1)$ -consistent*. A problem is *(strong) m -consistent* iff it is *(strong) $(m, 1)$ -consistent*. A problem is *path inverse consistent* (PIC) iff it is $(1, 2)$ -consistent.

A problem is *relationally arc consistent* (rel AC) iff any consistent instantiation for all but one of the variables in a constraint can be extended to the final variable so as to satisfy the constraint [12]. A problem is *strongly relationally arc consistent* (strong rel AC) iff any consistent instantiation of a subset of the variables in a constraint

¹ Department of Information & Communication Systems Engineering University of the Aegean, Greece.

² National ICT Australia & University of New South Wales, Australia.

can be extended to all the variables in the constraint. A problem is *relationally* (i, m)-consistent iff any consistent instantiation for i of the variables in a set of m constraints can be extended to all the variables in the set. We can construct singleton versions of all the relational consistencies in a straightforward manner [6]. For example, a problem is *relationally singleton arc consistent* (rel SAC) iff it has non-empty domains and for any instantiation of a variable, the resulting subproblem can be made relationally AC. A problem is *pairwise consistent* (PWC) iff it has non-empty relations and any consistent tuple in a constraint c can be consistently extended to any other constraint that intersects with c [9]. As shown in [9], applying PWC to a non-binary CSP is equivalent to applying AC to the dual encoding.

Following [6], we call a consistency property A stronger than B iff in any problem in which A holds then B holds, and strictly stronger (written $A \rightarrow B$) iff it is stronger and there is at least one problem in which B holds but A does not. We call a local consistency property A incomparable with B (written $A \otimes B$) iff A is not stronger than B nor vice versa. Finally, we call a local consistency property A equivalent to B (written $A \leftrightarrow B$) iff A is stronger than B and vice versa.

3 INVERSE CONSISTENCIES

In practice, most strong local consistency techniques have prohibitive space and time complexities. One way around this problem is to use inverse consistencies [8]. These require limited space as they only prune domains. When an inverse local consistency is enforced, it removes from the domain of a variable the values that cannot be consistently extended to some additional variables. For example, when enforcing PIC we remove values that cannot be consistently extended to any set of two other variables. By analogy with the definition of PIC, relational (1, 2) consistency is called *relational path inverse consistency* (rel PIC). We also define *pairwise inverse consistency* (PWIC), an inverse version of PWC. A value $a \in D(x_j)$ is PWIC iff \forall constraints c_i , where $x_j \in var(c_i)$, $\exists \tau \in rel(c_i)$ such that $\tau[x_j] = a$ and τ is valid and $\forall c_l$, s.t. $var(c_i) \cap var(c_l) \neq \emptyset$, $\exists \tau' \in rel(c_l)$, s.t. $\tau'[var(c_i) \cap var(c_l)] = \tau'[var(c_i) \cap var(c_j)]$ and τ' is valid. In this case we say that τ' is a PW-support of τ . A constraint c_i is PWIC iff $\forall x_j \in var(c_i)$, $\forall a \in D(x_j)$, a is PWIC. A problem is PWIC iff domains are non-empty and all constraints are PWIC.

We first compare rel PIC and PWIC with GAC, the most popular local consistency for non-binary constraints.

Theorem 1 $PWIC \rightarrow rel\ PIC \rightarrow GAC$

Proof: By definition, PWIC is stronger than rel PIC which is stronger than GAC. To show strictness, consider: $alldiff(x_1, x_2, x_3)$ and $x_1 = x_2$. If domains are $\{0, 1, 2\}$ then the problem is GAC but is not rel PIC. Now consider a constraint c_1 over variables x_1, x_2, x_3 and two other constraints c_2 and c_3 involving x_2 and x_3 (and other variables). Assume that value 0 of x_1 can be extended to tuples $(0, 0, 0)$ and $(0, 1, 1)$ in c_1 but only the first tuple can be extended to a consistent tuple in c_2 , while only the second tuple can be extended to a consistent tuple in c_3 . The problem is rel PIC but not PWIC. \square

Not surprisingly, when all constraints intersect on at most one variable, PWIC and rel PIC collapse down to GAC.

Theorem 2 *On constraints that intersect on at most one variable:*

$$PWIC \leftrightarrow rel\ PIC \leftrightarrow GAC$$

Proof: Suppose that the constraints intersect on at most one variable and are GAC. Consider an assignment for a variable and the set of constraints involving this variable. Take one of these constraints. As this is GAC, we can find a satisfying tuple including this assignment.

Consider all the intersecting constraints and the value of the intersection variable in our tuple. As all these constraints are GAC, we can extend the tuple to satisfy them. Hence, we can extend any assignment to a tuple that can be extended to satisfy all intersecting constraints. The problem is thus PWIC (and hence rel PIC). \square

When restricted to binary constraints, PWIC collapses down to rel PIC. We might expect rel PIC to reduce to the corresponding variable based definition. Surprisingly, this is not the case.

Theorem 3 *On binary constraints:*

$$\begin{array}{ccc} PWIC & \leftrightarrow & rel\ PIC \rightarrow rel\ AC \\ \uparrow & & \uparrow \\ PIC & \rightarrow & AC \end{array}$$

Proof: We first show $PWIC \leftrightarrow rel\ PIC$. Since binary constraints intersect on at most one variable unless they involve the same two variables, consider such a case with two variables x_1, x_2 , where all constraints are rel PIC. Take any two constraints. Since the constraints are rel PIC, any instantiation (x_1, a) can be extended to an instantiation (x_2, b) that satisfies both constraints. Since this holds for all pairs of constraints that involve x_1 and x_2 , these instantiations are consistent with all such constraints. Therefore, the problem is PWIC.

To show $PIC \rightarrow rel\ PIC$, consider the constraints: $x_1 \neq x_2, x_1 \neq x_3$ and $x_2 \neq x_3$. If all variables are 0/1 then the problem is rel PIC but not PIC. The other relations follow in a straightforward way. \square

Non-binary constraints can sometimes be decomposed into binary constraints on the same variables. For example, an all-different constraint can be decomposed into binary not-equals constraints. On such constraints, we can directly compare relational consistencies with binary consistencies on the decomposition.

Theorem 4 *On decomposable non-binary constraints:*

$$\begin{array}{c} \text{strong PC} \\ \otimes \quad \otimes \\ PWIC \rightarrow rel\ PIC \rightarrow AC \\ \otimes \quad \otimes \quad \otimes \quad \otimes \\ SAC\ PIC \quad SAC\ PIC \end{array}$$

Proof: To show $rel\ PIC \rightarrow AC$ on the decomposition, consider a problem that is rel PIC. Given any two constraints, any instantiation for a variable can be extended to all the variables in the two constraints. Hence, it can be extended to at least one other variable. Thus, the decomposition is AC. To show strictness, consider the constraints: $alldiff(x_1, x_2, x_3)$ and $x_3 = x_4$. If all variables are 0/1, then the problem is not rel PIC but its decomposition is AC.

To show the other relationships, consider the constraints: $alldiff(x_1, x_2, x_3, x_4)$ and $x_4 = x_5$. If domains are $\{0, 1, 2\}$, then the problem is not rel PIC (and hence not PWIC). However, its decomposition is strong PC (and thus SAC and PIC). For the reverse, consider the constraints: $x_1 \neq x_2, x_1 \neq x_3$, and $x_2 \neq x_3$. If all variables are 0/1, then the problem is rel PIC and PWIC, but its decomposition is not PIC (and hence not SAC nor strong PC). \square

One way to deal with non-binary constraints is to encode them into binary ones, and apply binary techniques (as for example in [1]). We now position rel PIC and PWIC with respect to other relational consistencies and consistencies enforced in the hidden variable and dual encodings of a non-binary problem. Each result has the precondition that the non-binary constraints are GAC so that the hidden variable or dual encoding is node consistent (and not trivially unsatisfiable).

Theorem 5 *On (non-binary) constraints which are GAC:*

$$\begin{array}{ccccc} SAC_{\text{hidden}} & \rightarrow & PIC_{\text{hidden}} & \leftrightarrow & AC_{\text{hidden}} \\ \downarrow & & \uparrow & & \\ rel\ SAC & \rightarrow & rel\ PIC & \otimes & (strong)\ rel\ AC \\ \uparrow & & \uparrow & & \otimes \\ SAC_{\text{dual}} & \rightarrow & PIC_{\text{dual}} & \rightarrow & AC_{\text{dual}} \end{array}$$

$$(PWC) \leftrightarrow PWIC$$

Proof: Due to space limitations we only give proofs of $\text{AC}_{\text{dual}} \leftrightarrow \text{PWIC}$, $\text{PIC}_{\text{dual}} \rightarrow \text{rel PIC}$ and $\text{rel PIC} \otimes (\text{strong}) \text{ rel AC}$.

To compare PWIC to AC_{dual} (and PWC), we only consider the values that are pruned by these consistencies. For AC_{dual} (and PWC), which delete tuples from constraints, this can be done if GAC is applied as a second step. In this case, a value is deleted iff all the tuples in some dual variable x , that include that value, are deleted. A tuple is deleted iff it has no support in some other dual variable (i.e. it is not pairwise consistent). In the non-binary problem, PWIC deletes the same values because it finds that all their extensions (i.e. tuples) in the constraint corresponding to x are not pairwise consistent. Therefore PWIC achieves the same pruning as AC_{dual} .

To show $\text{PIC}_{\text{dual}} \rightarrow \text{rel PIC}$, consider a problem whose dual encoding is PIC. Take any pair of constraints. There are two cases. In the first case, the two constraints are disjoint. As each constraint is GAC, we can extend any assignment for a variable in one of the constraints to satisfy all the variables in both constraints. In the second case, the two constraints overlap. As each constraint is GAC, we can extend any assignment for a variable in one of the constraints to satisfy all the variables in the constraint. As the dual encoding is PIC, we can extend this assignment for the dual variable associated with this constraint, to the dual variable associated with the second constraint, and any other third dual variable. That is, we can extend the assignment to satisfy all the variable in both the constraints. In both cases, the problem is rel PIC. To show strictness, consider the constraints: $x_1 \neq x_2$, $x_1 \neq x_3$ and $x_2 \neq x_3$. If all variables are 0/1 then the dual encoding is not PIC, whilst the original problem is rel PIC.

To show $\text{rel PIC} \otimes (\text{strong}) \text{ rel AC}$, consider the constraints: $\text{alldiff}(x_1, x_2, x_3)$ and $x_1 = x_2$. If domains are $\{0, 1, 2\}$ then the problem is (strong) rel AC but is not rel PIC. For the reverse, consider $\text{alldiff}(x_1, x_2, x_3)$. If domains are $\{0, 1, 2\}$, then the problem is not (strong) rel AC, but it is rel PIC. \square

One surprise here is that rel PIC is incomparable to (strong) rel AC whilst the binary local consistency PIC is strictly stronger than AC.

4 ALGORITHMS FOR PWIC

The generic AC-7 based algorithm for inverse local consistencies proposed in [13] can easily be adapted to enforce rel PIC. However, adapting the generic algorithm to enforce PWIC is more involved. The time and space complexities of this algorithm for rel PIC are $O(e^2kd^{2k})$ and $O(e^2k^2d)$ respectively, while for PWIC a naive adaptation would result in a very inefficient algorithm, in terms of time complexity. The time complexity of the PWC algorithm given in [9] is $O(e^2kd^{2k})$. This algorithm enforces PWC by applying AC in the dual encoding of the problem. Since PWC does not prune values from domains but instead deletes tuples from constraint relations, domains can be pruned if GAC is applied as a second step [9]. Apart from its high time complexity, this pruning method also suffers from a high space complexity, since constraints need to be extensionally represented. The time complexity of PWC can be reduced to $O(e^3d^k)$ if the algorithm of [11] for AC in the dual encoding is used. However, the space complexity can still be prohibitive.

In what follows we describe two algorithms for PWIC, which is stronger and more efficient than rel PIC, with better time complexity than a generic algorithm. These algorithms can be easily modified to achieve rel PIC. Our algorithms achieve the same pruning as a two-step procedure of PWC followed by GAC with better time and space complexity.

4.1 PWIC-1: A Simple Algorithm for PWIC

From the definition of PWIC we can immediately derive a simple algorithm by extending a GAC algorithm so that whenever it finds a GAC-supporting tuple for a value in a constraint c_i , it also checks if this tuple has a PW-support in all constraints intersecting with c_i . Figure 1 gives PWIC-1, an algorithm for PWIC based on the GAC algorithm GAC2001/3.1 [5].

```

function PWIC-1
1:    $Q \leftarrow \emptyset$ ;
2:   for each constraint  $c_i$ 
3:     for each variable  $x_j$  where  $x_j \in \text{var}(c_i)$ 
4:       if  $\text{Revise}(x_j, c_i) > 0$ 
5:         if  $D(x_j)$  is empty return INCONSISTENCY;
6:         put in  $Q$  each constraint  $c_m$  such that  $x_j \in \text{var}(c_m)$ ;
7:   return Propagation;
function Propagation
8:   while  $Q$  is not empty
9:     pop constraint  $c_i$  from  $Q$ ;
10:    for each unassigned variable  $x_j$  where  $x_j \in \text{var}(c_i)$ 
11:      if  $\text{Revise}(x_j, c_i) > 0$ 
12:        if  $D(x_j)$  is empty return INCONSISTENCY;
13:        put in  $Q$  each  $c_m \neq c_i$  such that  $x_j \in \text{var}(c_m)$ ;
14:   return CONSISTENCY;
function Revise( $x_j, c_i$ )
15:   for each value  $a \in D(x_j)$ 
16:     PW  $\leftarrow$  FALSE;
17:     for each valid  $\tau(\in \text{rel}(c_i)) \geq_l \text{lastGAC}_{x_j, a, c_i}$ 
           such that  $\tau[x_j] = a$ 
18:     PW  $\leftarrow$  TRUE;
19:     for each  $c_m$  such that  $\text{var}(c_i) \cap \text{var}(c_m) > 1$ 
20:       if  $\nexists$  valid  $\tau'(\in \text{rel}(c_m))$  such that
            $\tau[\text{var}(c_i) \cap \text{var}(c_m)] = \tau'[\text{var}(c_i) \cap \text{var}(c_m)]$ 
21:       PW  $\leftarrow$  FALSE; break;
22:     if PW=TRUE  $\text{lastGAC}_{x_j, a, c_i} \leftarrow \tau$ ; break;
23:     if PW=FALSE remove  $a$  from  $D(x_j)$ ;
24:   return number of deleted values;

```

Figure 1. A simple algorithm for PWIC.

Algorithm PWIC-1 uses a stack (or queue) of constraints to propagate value deletions, and works as follows. In the initial phase it iterates over each constraint c_i (line 2) and updates the domain of every variable x_j involved in c_i by calling Revise (line 4). During each revision, for each value a of $D(x_j)$ we first look for a tuple in $\text{rel}(c_i)$ that GAC-supports it. As in GAC2001/3.1, we store a pointer $\text{lastGAC}_{x_j, a, c_i}$. This is now the most recently discovered tuple in $\text{rel}(c_i)$ that GAC-supports value a of variable x_j and is pairwise consistent. If this tuple is valid then we know that a is GAC-supported. Otherwise, we look for a new GAC-support starting from the tuple immediately “after” $\text{lastGAC}_{x_j, a, c_i}$ (line 17). If $\text{lastGAC}_{x_j, a, c_i}$ is valid or a new GAC-support is found then the algorithm checks if the GAC-support (tuple τ) is pairwise consistent. Note that this check must be performed in the case where $\text{lastGAC}_{x_j, a, c_i}$ is valid, since this tuple may have lost its PW-supports on some of c_i ’s intersecting constraints.

To check if τ has PW-supports, PWIC-1 iterates over each constraint c_m that intersects with c_i on more than one variable. Constraints intersecting on one variable are not considered because PWIC offers here no more pruning than GAC. It checks if there is a tuple $\tau' \in \text{rel}(c_m)$ such that τ' is a PW-support of τ (lines

19-20). If such tuples are found for all intersecting constraints then $lastGAC_{x_j, a, c_i}$ is updated (line 22). If no PW-support is found on some intersecting constraint, then the iteration stops (line 21) and the algorithm looks for a new GAC-support. If no pairwise consistent GAC-support is found, a is removed from $D(x_j)$ (line 23). In this case, all constraints involving $D(x_j)$ not already in the stack are put on the stack (line 6). Constraints are then removed from the stack sequentially and their variables are revised. The algorithm terminates if a domain is wiped out, in which case the problem is not PWIC, or if the stack becomes empty, in which case the problem is PWIC.

Proposition 1 The worst-case time complexity of algorithm PWIC-1 is $O(e^2 k^3 d^{2k-f})$.

Proof: The complexity is determined by the *revise* function. $\text{Revise}(x_j, c_i)$ can be called at most kd times for each constraint c_i ; once for every deletion of a value from $D(x_j)$, where x_j is one of the k variables in $\text{var}(c_i)$. In each call, the algorithm performs one check to see if $\text{lastGAC}_{x_j, a, c_i}$ is valid. If $\text{lastGAC}_{x_j, a, c_i}$ is not valid, it tries to find a new GAC-support for a in $\text{rel}(c_i)$. The cost to make a GAC for the total number of calls to $\text{Revise}(x_j, c_i)$ is $O(kd^{k-1})$ (see [5] for details).

For each GAC-support τ found, PWIC-1 iterates over the, at most e , constraints that intersect with c_i to determine if τ has PW-supports. For each such constraint c_m , it checks at most d^{k-f} tuples, i.e. those that take the same values in variables $\text{var}(c_i) \cap \text{var}(c_m)$ as in τ . The cost of each such check is $O(k - f)$ if we assume this is linear in the arity of the tuple. Therefore, for each value of x_j , *Revise* makes $O(kd^{k-1} \times e(k - f)d^{k-f})$ checks. For kd values, the upper bound in checks performed to make one constraint PWIC is $O(ek^3d^{2k-f})$. For e constraints the complexity is $O(e^2k^3d^{2k-f})$. \square

The space complexity of PWIC-1 is $O(ekd)$.

4.2 PWIC-2: An Improved Algorithm for PWIC

Although the asymptotic time complexity of PWIC-1 is lower than that of the generic algorithm of [13], it can still be prohibitive in practice. PWIC-2 offers a significant improvement in terms of time complexity, but with an increase in the space complexity. The major bottleneck for PWIC-1 is that each time a GAC-support τ for a value $a \in D(x_j)$ is found in $rel(c_i)$, it has to check if τ is pairwise consistent. This is done by iterating through all constraints that intersect with c_i . In each such iteration the algorithm may check **all** the d^{k-f} sub-tuples that include the assignment $\tau[var(c_i) \cap var(c_m)]$. This process is repeated each time c_i is revised. To overcome this problem, for each constraint c_i algorithm PWIC-2 keeps a set of d^f pointers for every constraint c_m intersecting with c_i . Each such pointer $lastPW_{c_i, c_m, s}$ corresponds to the s -th sub-tuple among the d^f sub-tuples containing the possible combinations of values for variables $var(c_i) \cap var(c_m)$. Each pointer points to the most recently discovered tuple in $rel(c_m)$ that extends sub-tuple s and is a PW-support for the current GAC-support of a value (x_j, a) on c_i .

Figure 2 gives lines 20-21 of function Revise of PWIC-2 (the rest of the algorithm is the same as PWIC-1). During each revision, for each value a of $D(x_j)$ we first look for a tuple in $rel(c_i)$ that GAC-supports it, in the same way as in PWIC-1. If such a tuple τ is found then the algorithm checks if τ is pairwise consistent. For each constraint c_m that intersects with c_i PWIC-2 first checks if tuple $lastPW_{c_i, c_m, s}$ is valid, where s is the sub-tuple $\tau[\text{var}(c_i) \cap \text{var}(c_m)]$. If it is valid then it PW-supports τ since they have the same values for the variables $\text{var}(c_i) \cap \text{var}(c_m)$. Otherwise, the algorithm looks for a new PW-support starting from the tuple that

```

function Revise( $x_j, c_i$ )
     $s \leftarrow \tau [var(c_i) \cap var(c_m)];$ 
    if  $lastPW_{c_i, c_m, s}$  is not valid
        if  $\exists$  valid  $\tau' (\in rel(c_m)) >_l lastPW_{c_i, c_m, s}$ 
            and  $\tau' [var(c_i) \cap var(c_m)] = s$ 
             $lastPW_{c_i, c_m, s} \leftarrow \tau';$ 
    else PW  $\leftarrow$  FALSE; break;

```

Figure 2. Function Revise of PWIC-2.

includes the assignment s and is immediately “after” $\text{lastPW}_{c_i, c_m, s}$ in the lexicographic order. If such a tuple τ' is found, $\text{lastPW}_{c_i, c_m, s}$ is updated. If no tuple is found in $\text{rel}(c_i)$ that is both a GAC-support for a and is pairwise consistent, then a is removed from $D(x_j)$.

The following example demonstrates the savings in constraint checks that PWIC-2 achieves compared to PWIC-1.

Example 1 Consider the problem of Figure 3. There are four variables $\{x_1, \dots, x_4\}$ and two constraints that intersect on x_1 and x_2 . Tuples in italics are inconsistent and the rest are consistent. Assume we wish to determine if the values for x_3 are PWIC. PWIC-1 checks all 5 tuples of c_2 for each value of x_3 , as depicted in Figure 3a (for each tuple τ in c_1 , all tuples of c_2 between the pair of edges starting from τ are checked against τ). PWIC-2 checks all 5 tuples only for value 0 of x_3 . After locating $\{0, 0, 4\}$ of c_2 as a PW-support for $\{0, 0, 0\}$ of c_1 , $lastPW_{c_1, c_2, s}$, where $s = \{0, 0\}$, points to $\{0, 0, 4\}$. For the rest of x_3 's values, PWIC-2 only checks this tuple.

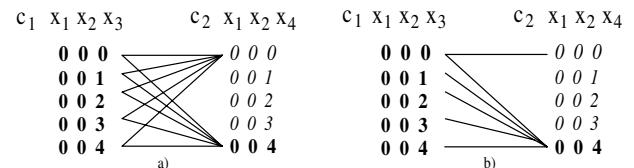


Figure 3. Applying PWIC-1 and PWIC-2.

Proposition 2 The worst-case time complexity of algorithm PWIC-2 is $O(e^2 k^2 d^k)$.

Proof: The complexity is again determined by the calls to `Revise`. When looking for a GAC-support within `Revise`, PWIC-2 is identical to PWIC-1 and therefore the two algorithms have the same cost for this part. That is, $O(kd^{k-1})$ to make a value $a \in D(x_j)$ GAC for all calls to `Revise`(x_j, c_i). Once a GAC-support τ is found, PWIC-2 iterates over the constraints that intersect with c_i . Assuming that $\tau[\text{var}(c_i) \cap \text{var}(c_m)] = s$, for any intersecting constraint c_m PWIC-2 searches through the tuples that include assignment s . However, these tuples are not searched from scratch every time. Since the pointer $\text{lastPW}_{c_i,c_m,s}$ is used, the tuples that include assignment s are searched from scratch **only** during the first time a GAC-support τ with $\tau[\text{var}(c_i) \cap \text{var}(c_m)] = s$ is located. In any subsequent iteration of the for loop in line 17 of Figure 1, and in any subsequent call to `Revise`(x_j, c_i), whenever a GAC-support that includes s is located, the algorithm first checks if $\text{lastPW}_{c_i,c_m,s}$ is still valid. If it is not, then only tuples “after” $\text{lastPW}_{c_i,c_m,s}$ are searched. As a result, in the $O(d^{k-1})$ times a GAC-support for a is located, each

tuple of each intersecting constraint is checked at most once (with each check costing $O(k - f)$) and there are at most d^{k-1} checks for the validity of $\text{last } PW_{c_i, c_m, s}$. Thus, the cost of $\text{Revise}(x_j, c_i)$ is $O(kd^{k-1} + e(d^{k-1} + (k-f)(d^{k-1}))) = O(ekd^{k-1})$. For the $O(kd)$ times Revise is called to make a constraint PWIC, the upper bound in checks is $O(ek^2d^k)$. For e constraints the worst-case complexity is $O(e^2k^2d^k)$. \square

The space complexity of PWIC-2 is $O(e^2d^f)$. PWIC-2 is thus not practical for large arity constraints sharing many variables. Note that, even when constraints share just two variables (and the space complexity is $O(e^2d^2)$), PWIC is stronger than GAC. It may be beneficial to apply PWIC only to variables participating in certain constraints, based on properties like the arity and the number of shared variables, and to apply GAC to the rest of the variables. Indeed, our two algorithms for PWIC apply such a hybrid propagation scheme, as they apply GAC to constraints that intersect on just one variable.

5 EXPERIMENTS

We compared algorithms that maintain GAC2001/3.1, PWIC-1, PWIC-2, RPIC-1, and RPIC-2 throughout search (RPIC-1 and RPIC-2 are algorithms for rel PIC similar to the corresponding algorithms for PWIC). We simply refer to these search algorithms by the consistency enforced. All algorithms used the dom/deg variable ordering heuristic [4]. Figure 4 shows the average CPU time to solve 50 instances of randomly generated problems with 30 variables, uniform domain size of 10, 28 4-ary constraints, and varying constraint looseness (along the x-axis). The problems were generated using the extended model B [2]. On these instances, PWIC-1 and PWIC-2 are faster than GAC and RPIC-1 and RPIC-2 are slower. From experiments with other parameters, we conjecture that PWIC is more efficient than GAC on sparse problems, especially when the domain size is large. On the other hand, PWIC is too expensive on problems with medium or high density. Although it significantly reduces the nodes visited, it is outperformed by GAC in CPU time. Rel PIC is generally less efficient than both GAC and PWIC.

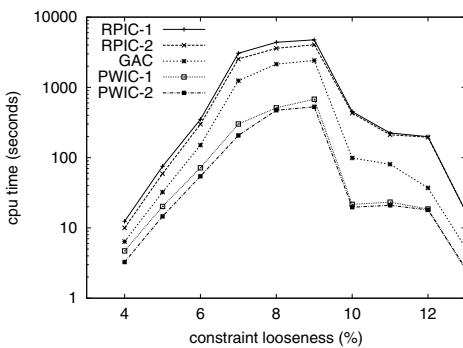


Figure 4. Comparison on random problems.

We also ran experiments on the CLib configuration benchmark library (see www.it.u.edu/VeCoS/clib). To obtain hard instances, each problem was slightly altered by adding a few variables (5-6) and constraints (8-10) randomly, as described in [11].

Table 1 gives results for GAC, and (for space reasons) only PWIC-1 and RPIC-1. There is a significant difference in run times in favor of the algorithms that maintain inverse consistencies due to the additional pruning they perform. PWIC-2 and RPIC-2 are faster than

problem	n	e	k	d	GAC nodes-time	PWIC-1 nodes-time	RPIC-1 nodes-time
machine	30	22	4	9	535874-14,38	12600-1,31	36046-4,27
fx	24	21	5	44	193372-4,06	1315-0,09	17624-1,32
fs	29	18	6	51	618654-23,29	19-0,00	1698-0,23
esvs	33	20	5	61	6179966-153,71	7859-0,24	39021-3,21

Table 1. Configuration problems. k and d are the maximum arity and domain size. Averages are over 50 hard instances for each benchmark.

PWIC-1 and RPIC-1, by a small margin on average. The PWIC algorithms are again more efficient than the rel PIC ones.

6 CONCLUSION

Although domain filtering consistencies tend to be more practical than consistencies that change the constraint relations and the constraint graph, very few such consistencies have been proposed for non-binary constraints. In this paper, we performed a detailed study of two such consistencies, rel PIC and PWIC. Our theoretical study revealed some surprising results. For example, rel PIC and PWIC are weaker than PIC when restricted to binary constraints, while for non-binary constraints rel PIC and PWIC are incomparable to rel AC. We also described algorithms that can be used to achieve PWIC and rel PIC. One has a particularly good time complexity, competitive with GAC algorithms, though with a higher space cost. Experiments demonstrated the potential of inverse consistencies as an alternative or complementary to GAC. As future work, we will investigate ways to combine inverse consistencies with specialized GAC propagators.

ACKNOWLEDGEMENTS

We would like to thank Patrick Prosser for useful comments and suggestions on an early draft of the paper.

REFERENCES

- [1] F. Bacchus, X. Chen, P. van Beek, and T. Walsh, ‘Binary vs. Non-binary CSPs’, *Artificial Intelligence*, **140**, 1–37, (2002).
- [2] C. Bessière, P. Meseguer, E.C. Freuder, and J. Larrosa, ‘On Forward Checking for Non-binary Constraint Satisfaction’, *Artificial Intelligence*, **141**, 205–224, (2002).
- [3] C. Bessière and J.C. Régin, ‘Arc Consistency for General Constraint Networks: Preliminary Results’, in *Proc. of IJCAI’97*, pp. 398–404, (1997).
- [4] C. Bessière and J.C. Régin, ‘MAC and Combined Heuristics: Two Reasons to Forsake FC (and CBJ?) on Hard Problems’, in *Proc. of CP’96*, pp. 61–75, (1996).
- [5] C. Bessière, J.C. Régin, R. Yap, and Y. Zhang, ‘An Optimal Coarse-grained Arc Consistency Algorithm’, *Artificial Intelligence*, **165**(2), 165–185, (2005).
- [6] R. Debruyne and C. Bessière, ‘Domain Filtering Consistencies’, *JAIR*, **14**, 205–230, (2001).
- [7] E. Freuder, ‘A Sufficient Condition for Backtrack-bounded Search’, *JACM*, **32**(4), 755–761, (1985).
- [8] E. Freuder and C. Elfe, ‘Neighborhood Inverse Consistency Preprocessing’, in *Proc. of AAAI’96*, pp. 202–208, (1996).
- [9] P. Janssen, P. Jégou, B. Nouguier, and M.C. Vilarem, ‘A filtering process for general constraint satisfaction problems: Achieving pairwise consistency using an associated binary representation’, in *Proc. of IEEE Workshop on Tools for Artificial Intelligence*, pp. 420–427, (1989).
- [10] P. Jégou, ‘On the Consistency of General Constraint Satisfaction Problems’, in *Proc. of AAAI’93*, pp. 114–119, (1993).
- [11] N. Samaras and K. Stergiou, ‘Binary Encodings of Non-binary CSPs: Algorithms and Experimental Results’, *JAIR*, **24**, 641–684, (2005).
- [12] P. van Beek and R. Dechter, ‘On the Minimality and Global Consistency of Row-convex Constraint Networks’, *JACM*, **42**(3), 543–561, (1995).
- [13] G. Verfaillie, D. Martinez, and C. Bessière, ‘A Generic Customizable Framework for Inverse Local Consistency’, in *Proc. of AAAI’99*, pp. 169–174, (1999).

Guiding Search using Constraint-level Advice

Radoslaw Szymanek and Barry O'Sullivan¹

Abstract. Constraint satisfaction problems are traditionally solved using some form of backtrack search that propagates constraints after each decision is made. The efficiency of search relies heavily on the use of good variable and value ordering heuristics. In this paper we show that constraints can also be used to guide the search process by actively proposing the next choice point to be branched on. We show that search effort can be reduced significantly.

1 INTRODUCTION

Constraint programming (CP) has become one of the dominant approaches to modelling and solving real-world combinatorial problems [19]. However, while CP has many success stories, it is believed that improving the usability of the technology is a key factor in its future success [15]. This is largely due to the fact that using constraints technology often requires considerable expertise in the use of constraint programming tools. It has been argued [15] that the constraints community should focus on developing techniques that do not require the user to tweak the search algorithm, for example by selecting variable and value ordering heuristics. Instead the solver should automatically configure itself based on the problem formulation. It is this need that motivates the work we present in this paper.

The objective of this work is to show that constraints can be used to efficiently guide search. Our results show that search effort can be reduced significantly. We see the major contribution of this paper being a step towards an approach to CP where the constraint solver dynamically modifies its behaviour in response to the particular formulation of the problem, specifically the semantics of the constraints among variables rather than simply the existence of them. We demonstrate this by showing that the functionality of global constraints can be augmented so they can successfully guide search.

The remainder of this paper is organised as follows. Section 2 presents the framework for constraint-level guiding. An example of how constraints can guide search is presented in Section 3. A detailed empirical evaluation is presented in Section 4. In Section 5 we relate our work to the relevant literature. We outline some directions for future work and make some concluding remarks in Section 6.

2 THE FRAMEWORK

A constraint satisfaction problem (CSP) is a 3-tuple $P \doteq \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where \mathcal{X} is a finite set of variables $\mathcal{X} \doteq \{x_1, \dots, x_n\}$, \mathcal{D} is a set of finite domains $\mathcal{D} \doteq \{D(x_1), \dots, D(x_n)\}$ where the domain $D(x_i)$ is the finite set of values that variable x_i can take, and a set of constraints $\mathcal{C} \doteq \{c_1, \dots, c_m\}$. Each constraint c_i is defined by the ordered set $\text{var}(c_i)$ of the variables it involves, and a set $\text{sol}(c_i)$ of

allowed combinations of values. An assignment of values to the variables in $\text{var}(c_i)$ satisfies c_i if it belongs to $\text{sol}(c_i)$. A solution to a CSP is an assignment to each variable with a value from its domain such that every constraint in \mathcal{C} is satisfied.

Many modern constraint solvers by default use a 2-way, or binary branching, search strategy [13, 18]. When a variable x is selected for instantiation, a value is chosen to generate a binary choice point. For example, if value v is selected, the corresponding choice point (x, v) creates two branches, $x = v$ and $x \neq v$. The first (the left branch) is explored; if that branch fails, the search backtracks, and the right branch is followed instead by propagating $x \neq v$. In general, if a dynamic variable ordering is being used, the choice of variable may change after propagating $x \neq v$ (right branch).

For each choice point we can use individual constraints to estimate the amount of pruning that can be achieved in each branch in a number of ways. We assume that each constraint $c_i \in \mathcal{C}$ has an associated function $\mathbf{g}_i(\cdot)$ that, when given a choice point cp as an argument², computes a pair of values $(\min_{cp}^{c_i}, \max_{cp}^{c_i})$, where $\min_{cp}^{c_i}$ (resp. $\max_{cp}^{c_i}$) is an estimate of the minimum (resp. maximum) amount of pruning associated with the choice point cp by propagating constraint c_i , e.g. measured in terms of the number of values removed from the domains of the variables in $\text{var}(c_i)$. There are many ways of defining the function $\mathbf{g}_i(\cdot)$ depending on the type of constraint being considered. For example, in the case of an ALLDIFFERENT or PERMUTATION constraint [17], the estimate of pruning that can be achieved from a choice point can be based on the number of arcs removed from the value graph maintained by the constraint. Similarly, in the case of an extensionally represented constraint the estimate can be based on the number of lost supports associated with both branches of the choice point; in the case of the SUM constraint [1], which achieves bounds consistency, it can be based on how much the bounds are tightened.

We can compare the quality of choice points based on their respective pruning estimates using a comparison operator \succ , which induces a total order over choice points. There are many ways to define the semantics of \succ . For example, given two choice points cp_a, cp_b and a constraint c_i we might say that $\mathbf{g}_i(cp_a) \succ \mathbf{g}_i(cp_b) \leftrightarrow \min_{cp_a}^{c_i} > \min_{cp_b}^{c_i}$, i.e. that we consider cp_a to be better than cp_b with respect to constraint c_i iff the minimum amount of pruning associated with cp_a is larger than that of cp_b . It is also useful to compare pruning due to different constraints, i.e. comparing $\mathbf{g}_i(cp_a) \succ \mathbf{g}_j(cp_b)$.

In order to use constraints to guide search, our approach is to firstly *use each constraint to evaluate the default choice point* proposed by the default variable and value orderings used in the search algorithm. Formally, consider a choice point $cp_0 = (x, v)$ proposed by the default search orderings. The best pruning estimate, $\text{eval}(cp_0)$, for this choice point is computed by $\mathbf{g}_i(cp_0)$ such that

¹ Cork Constraint Computation Centre, University College Cork, Ireland.
 {r.szymanek|b.osullivan}@4c.ucc.ie

² Implicitly \mathbf{g}_i also takes a second argument representing the current state of the domains of the variables, but we omit this to keep our notation simpler.

$\neg \exists c_j \in (\mathcal{C} - \{c_i\}).(\mathbf{g}_j(cp_0) \succ \mathbf{g}_i(cp_0))$. Secondly, once the best pruning estimate is obtained, we use each constraint in turn to propose the best choice point from its local perspective. If a constraint's local best choice point proposal is better, according to \succ , than the best proposed to date, we use it as the baseline when querying the next constraint. Formally, for each constraint c_i we can define the best choice point for that constraint, \mathbf{best}_i , as follows: let $\Delta = \{(x, v) | x \in \text{var}(c_i), v \in D(x)\}$, thus $\mathbf{best}_i = cp \in \Delta$ such that $\neg \exists cp' \in (\Delta - \{cp\}).\mathbf{g}_i(cp') \succ \mathbf{g}_i(cp)$.

The choice point we branch on is the best one found having considered each constraint in \mathcal{C} . Formally, we recursively consider each constraint $c_1 \dots c_m \in \mathcal{C}$ to compute the overall best choice point, cp_m , as follows:

$$cp_m = \begin{cases} cp_{m-1} & \mathbf{g}_k(cp_{m-1}) \succ \mathbf{g}_m(\mathbf{best}_m), \\ \mathbf{best}_m & \text{otherwise.} \end{cases}$$

where \mathbf{g}_k is the evaluation given to the choice point cp_{m-1} by the constraint c_k that proposed it; cp_0 is the choice point proposed by the default search orderings, and $\mathbf{g}_0(cp_0) = \text{eval}(cp_0)$.

In practice, this framework may be modified in two ways. Firstly, we may wish to use only a subset of the constraints \mathcal{C} to guide search. In this situation the remaining constraints are not considered in the evaluation and proposal of choice points. Secondly, if the user does not wish to define a default search strategy, the guiding constraints can guide search without firstly evaluating the default choice point, thus reducing the overhead associated with search guiding. The modifications to the description, and the algorithms presented later, are obvious.

3 AN EXAMPLE

We present an example of how constraints can be used to guide search. The example uses the PERMUTATION constraint to demonstrate how the internal data structures maintained by this global constraint can be used to evaluate and propose choice points. A PERMUTATION constraint is applicable when the number of variables and values is equal and we wish to ensure that each variable takes a different value. Therefore, the PERMUTATION constraint can be regarded as a special case of the ALLDIFFERENT constraint [17]. A filtering algorithm for PERMUTATION can be readily derived from the filtering algorithm for ALLDIFFERENT.

We will consider an example permutation problem: quasigroup completion (QCP). Essentially, a quasigroup can be viewed as an $n \times n$ multiplication table defining a Latin square, which must be filled with unique integers on each row and column. One possible model for a QCP of order n is to have n^2 variables, $x_1, \dots, x_{(n^2)}$, and specify a PERMUTATION constraint on each set of n variables representing each row and column. An example of QCP with four cells preassigned is presented in Figure 1. We will concentrate only on one of the permutation constraints in this model, which is imposed over variables from the top row, comprising variables x_1, \dots, x_4 . The PERMUTATION constraint maintains an internal data structure called the value graph to achieve generalised arc consistency (GAC) [17]. The value graph is a bipartite graph in which the edges link variables to values in their current domain. The state of the value graph after propagating all initial assignments is presented in Figure 1. This value graph can be reused to identify values that can be assigned to a small number of variables, as well as variables with small domain.

In order to implement a variant of the PERMUTATION constraint that can propose choice points, the constraint must be modified to

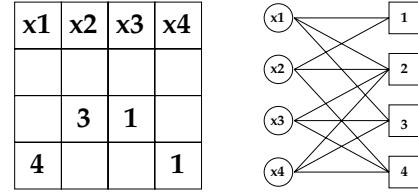


Figure 1. An example QCP and the PERMUTATION value graph.

evaluate multiple candidates. Evaluating every possible choice point may be too costly, so a heuristic can be used to reduce the set of candidates.

To compare choice points, and ultimately propose them based on constraint-level advice, we can consider a variety of heuristics. For example, in Figure 2 we present two candidate choice points. The first candidate uses variable x_1 and value 3. The pruning achieved in both branches of this choice point is depicted in Figure 2(a). The left branch gives a lot of pruning, unfortunately the right branch achieves the least amount of pruning possible. If we choose the left branch first and it does not lead to a solution, then the discovery of this mistake has negligible positive effect on the search. If we choose the right branch first and it does not lead to a solution, then the discovery of this may take long time, as little pruning is achieved. The second choice point uses variable x_2 and value 1. The pruning achieved by this choice point is depicted in Figure 2(b). This choice point offers an interesting trade-off. If the left branch does not lead to a solution, we still obtain a reasonable amount of pruning after backtracking from it. We might prefer this type of choice point because it tends to hedge against the possibility that we make a mistake when selecting the left branch.

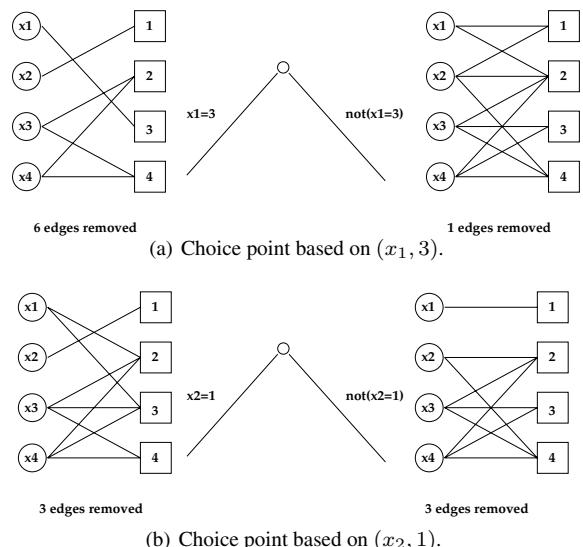


Figure 2. An comparison of alternative choice points.

It is also possible to obtain value guidance only. Given a variable selected by the default variable ordering heuristic, e.g. x_1 , then after the analysis of the value graph, value 1 for this variable might be proposed, for example, based on the fact that it can be assigned to the

fewest number of variables. Similarly, we can also obtain *variable guidance only*, using other external methods to decide on a value.

4 EMPIRICAL EVALUATION

We report on experiments conducted to verify the utility of our approach to constraint-guided search. Our experiments show that such guidance can be very successful in many problem domains such as QCP, Magic Squares and Langford's Problem. We describe our problem domains and the models in Section 4.1. In Section 4.2 we present the details of the constraint-level guidance mechanism used in the experiments. Our results are presented in Section 4.3.

4.1 Problem Domains and Models

The problem domains studied, taken from CSPLIB³, are as follows:

Quasigroup Completion. A quasigroup is a set Q with a binary operation $\star : Q \times Q \rightarrow Q$, such that for all a and b in Q there exist unique elements in Q such that $a \star x = b$ and $y \star a = b$. The cardinality of the set, $n = |Q|$, is called the order of the quasigroup. A quasigroup can be viewed as an $n \times n$ multiplication table defining a Latin square, which must be filled with unique integers on each row and column. The Quasigroup Completion Problem (QCP) is the problem of completing a partially filled Latin square.

Magic Squares. An order n magic square is a $n \times n$ matrix containing the numbers 1 to n^2 , with each row, column and the main diagonals summing up to the same number.

Generalization of Langford's Number Problem. An instance $L(k, n)$ of Langford's Number problem seeks to arrange k sets of numbers 1 to n , so that each appearance of the number m is m numbers from the last. For example, the $L(3, 9)$ problem is to arrange 3 sets of the numbers 1 to 9 so that the first two 1's and the second two 1's appear one number apart, the first two 2's and the second two 2's appear two numbers apart, etc. A solution to $L(2, 4)$ is 41312432.

All models and algorithms used in our evaluation were implemented using the JACOP constraint programming system [12]. Experiments were run on a Pentium 4 processor (1800 MHz), with 512Mb RAM running Linux Fedora Core 4. The models used for each problem were those from [11]. For each problem domain we compared the guided approach against both a primal model and a dual model. In both models the variable ordering heuristic used preferred variables with the smallest domain (ties broken lexicographically) with values selected lexicographically. However, for the dual model the set of search variables included both the primal and dual variables. Therefore, in the dual model the search heuristic was equivalent to the $SD(p + d)$ heuristic presented in [11]. The dual model differed from the primal model since it had dual variables and channelling constraints between primal and dual variables. The channelling constraints were implemented as one global constraint to increase the efficiency. There was no need to post PERMUTATION on the dual variables since PERMUTATION on the primal variables achieved GAC. Therefore, the dual model, and its heuristic, used in our experiments was the most efficient one studied in [11].

In addition to the guidance of the PERMUTATION constraint discussed in Section 3, we also implemented a simple form of guidance for the SUM constraint. Our implementation of the SUM constraint achieves bounds consistency. The choice point that tightens the bounds on the domains of the variables most is preferred.

³ <http://www.csplib.org>.

4.2 Implementation of Guidance

We implemented the guidance framework presented in Section 2 using Algorithm 1. The algorithm consists of an evaluation phase, in which the choice point proposed by the default heuristic is analysed to compute $\text{eval}(cp_0)$, and a proposal phase where the best choice point to be branched on is computed. The function $g_i(\cdot)$ is implemented using $\min(cp, c)$ and $\max(cp, c)$, which compute the minimal and maximal pruning obtained by constraint c given choice point cp , respectively, based on the heuristic estimate outlined earlier. The only constraints used to guide in our models were the PERMUTATION and SUM constraints.

Algorithm 1: GUIDEDCHOICE(cp_0, \mathcal{C})

```

Input: A choice point  $cp_0 = (var, val)$  where  $var$  is the variable proposed by the default variable ordering and  $val$  is the value proposed by the default value ordering.
 $\mathcal{C}$ , the constraints in the problem.
Output: A choice point,  $cp$ , considered best by the constraints in  $\mathcal{C}$ .
 $cp \leftarrow cp_0$ ;
 $\min_{cp} \leftarrow 1$ ;
 $\max_{cp} \leftarrow \infty$ ;
foreach constraint  $c_i$  in  $\mathcal{C}$  do
    if BETTER( $\min(cp_0, c_i), \max(cp_0, c_i), \min_{cp}, \max_{cp}$ ) then
         $\min_{cp} \leftarrow \min(cp_0, c_i)$ ;
         $\max_{cp} \leftarrow \max(cp_0, c_i)$ ;
foreach constraint  $c_i$  in  $\mathcal{C}$  do
     $\text{best}_i \leftarrow \text{PROPOSEBESTCHOICE}(c_i)$ ;
    if BETTER( $\min(\text{best}_i, c_i), \max(\text{best}_i, c_i), \min_{cp}, \max_{cp}$ ) then
         $cp \leftarrow \text{best}_i$ ;
         $\min_{cp} \leftarrow \min(\text{best}_i, c_i)$ ;
         $\max_{cp} \leftarrow \max(\text{best}_i, c_i)$ ;
return  $cp$ ;

```

The function PROPOSEBESTCHOICE returns the best choice point from the perspective of a guiding constraint, c_i , based on \succ . The comparison operator, \succ , was implemented using Algorithm 2. The first criterion used by \succ is to maximise the minimal amount of pruning obtained by a choice point, denoted \min_{cp_x} . If two choice points have the same amount of minimal pruning then a tie-breaker based on the maximal amount of pruning, \max_{cp_x} , is employed. The tie-breaker behaves differently depending on the minimal amount of pruning. In experiments this implementation of the \succ operator was found to be the most robust over all problem classes studied.

Algorithm 2: BETTER($\min_{cp_x}, \max_{cp_x}, \min_{cp_y}, \max_{cp_y}$)

```

Input : Estimates of minimal and maximal pruning of choice points,  $cp_x$  and  $cp_y$ .
Output: A boolean reflecting whether the pruning estimate for  $cp_x$  is considered better than the estimate for  $cp_y$ .
if ( $\min_{cp_x} > \min_{cp_y}$ ) then
    return TRUE;
else if ( $\min_{cp_x} = \min_{cp_y} = 1 \wedge \max_{cp_x} \leq \max_{cp_y}$ ) then
    return TRUE;
else if ( $\min_{cp_x} = \min_{cp_y} \neq 1 \wedge \max_{cp_x} \geq \max_{cp_y}$ ) then
    return TRUE;
else return FALSE;

```

4.3 Results and Discussion

In Table 1 we present results for the QCP. We used problems of order 20 with 187 holes to give us instances at the peak of difficulty for filtered QCP. We generated 3000 instances, using code based on Carla

Gomes' *Isencode* quasigroup generator, from which we filtered and retained only the satisfiable ones, giving us 1749 instances.

We present results for the dual model, the primal non-guided model, the primal model with value-only guiding, the primal model with variable-only guiding and the primal model with choice point (variable and value) guiding. With the exception of the dual model, whose heuristic has been discussed above, in each case we used the smallest-domain-first (sdf), random and largest-domain-first (i.e. anti-sdf) variable ordering heuristics. Values were chosen lexicographically. The model we used for these problems employed PERMUTATION constraints to ensure uniqueness of elements across columns and rows.

In each case we present the percentage of instances solved within a limit of 100000 backtracks. Any problems that exceeded this number of backtracks were removed from the data-set when computing our results. We also present the median number of nodes, median number of backtracks and median time required in seconds. These measurements were taken due to maladies suffered by each measure on its own and according to the definitions given in [3].

Table 1. Results for filtered QCP order 20 (guided highlighted).

model (heuristic)	% solved	nodes	backt.	time (s)
dual (sdf,sdf)	99.9	326	149	17.52
primal (sdf)	99.9	323	149	3.46
val guided (sdf)	99.9	244	105	2.77
var guided (sdf)	99.8	169	69	2.70
guided (sdf)	99.8	162	67	2.66
primal (random)	91.8	6,138	3,055	34.95
val guided (random)	98.7	1,926	946	12.06
var guided (random)	99.7	162	66	2.46
guided (random)	99.7	164	68	2.45
primal (anti-sdf)	76.4	14,184	7,068	57.96
val guided (anti-sdf)	97.1	4,069	2,017	19.18
var guided (anti-sdf)	99.7	165	68	2.40
guided (anti-sdf)	99.7	166	69	2.41

It is clear that a standard dual model does not help for this problem, and is outperformed by the primal (sdf) model. This is due to the structural properties of the QCP problem and the fact that PERMUTATION constraints enforcing GAC were used. It is clear that the additional variables, dual constraints and channelling constraints are simply adding overhead. Considering the primal, value guided, variable guided and (choice) guided cases, we note that as the sophistication of the type of guiding employed increases, we observe improvements in search effort, regardless of the default search heuristics used.

Comparing the primal (sdf) results with the guided (sdf) results it can be seen that guiding improves upon the already good variable ordering used by default. Clearly as search progresses, the guiding mechanism is capable of proposing alternative choice points that are of sufficiently high quality to improve overall performance.

When a poor default search heuristic is selected (random and anti-sdf) the guiding mechanism is strong enough to redirect search in a way that is essentially as good as the best search strategy observed in the experiment, i.e. guided (sdf). This is quite a significant result since it shows that the poor choice of heuristic is essentially irrelevant in this case. The guiding mechanism that has been incorporated into the global constraints used to model the problem is capable of compensating for the poor choice of heuristic. In a sense, the search strategy is being dynamically improved by knowledge that is available at little extra cost within the global constraints.

The primal model without guiding was able to solve two more instances comparing to some guided searches. This is reflected in the marginal difference in the percentage of solved instances. We looked at these two instances carefully and after analysing them we concluded that guided search in those two cases had chosen the wrong branch at the root of the tree and was not able to recover before hitting the cutoff limit. This simply emphasises the fact that even the best variable ordering can fall victim to bad value ordering. However, two instances are of no statistical significance.

Table 2. Results for Magic Squares (guided highlighted).

instance	model	nodes	backt.	time (s)
3	primal	4	0	0.22
3	dual	4	0	0.28
3	guided	4	0	0.28
4	primal	18	3	0.44
4	dual	11	0	0.45
4	guided	18	0	0.52
5	primal	1,365	658	2.23
5	dual	1,016	490	2.91
5	guided	814	375	2.11
6	primal	111,153	55,525	124.28
6	dual	—	—	—
6	guided	37,116	18,512	65.14

In the case of the Magic Squares (Table 2) we report results for three different approaches. The first one employs a primal model with the sdf heuristic. The second one is a primal model where all SUM constraints are guiding search. The guided search uses sdf as the default heuristic. Finally, the third approach uses a dual model and the $SD(p+d)$ heuristic. For small and easy problems the best approach is the primal model, as the worst case search space is small and cost per search node is the deciding factor. For larger orders, namely 5 and 6, constraint-guided search can significantly reduce the number of search nodes and can bring almost a 50% reduction in search time compared to the primal model. The dual model could not find any solution for order 6 as it reached the cutoff limit, which was increased to 500000 backtracks for these problems.

For the generalised Langford's problem, we only present results for instances that involved more than 1000 nodes to solve (Table 3). Again, it is clear that constraint-guided search is superior than solving using a traditional approach based on ordering heuristics only. We see improvements using all metrics (backtracks, nodes and time) when comparing the primal model with the guided one. We can also see that the guided model can be of benefit over a primal model, achieving a factor of 15 reduction in search time for the largest instance studied. The guiding approach, while not requiring us to formulate a dual model of our problem, out-performs the dual model in most cases in terms of time since the savings in nodes and backtracks that are gained by the dual model do not payoff when the overhead of the dual model is taken into consideration.

5 RELATED WORK

Variable and value ordering heuristics are the most common approaches to guiding search. Many variable ordering heuristics have been proposed in the literature [2, 5, 6, 10]. The disadvantage of variable ordering heuristics is that they are usually based on macro features of a CSP, such as the domain size of variables and the number

Table 3. Results for Langford's problem (guided highlighted).

instance	SAT?	model	nodes	backt.	time (s)
L(2,9)	n	primal	9,895	4,947	4.5
L(2,9)	n	dual	8,011	4,005	6.4
L(2,9)	n	guided	8,015	4,007	4.4
L(2,10)	n	primal	53,633	26,816	22.6
L(2,10)	n	dual	43,603	21,801	32.7
L(2,10)	n	guided	42,783	21,391	20.8
L(3,11)	n	primal	10,101	5,050	16.7
L(3,11)	n	dual	5,793	2,896	17.6
L(3,11)	n	guided	5,917	2,958	11.4
L(3,12)	n	primal	41,573	20,786	71.1
L(3,12)	n	dual	22,897	11,448	70.0
L(3,12)	n	guided	24,141	12,070	46.3
L(3,13)	n	primal	186,111	93,055	330.1
L(3,13)	n	dual	88,505	44,252	285.7
L(3,13)	n	guided	95,655	47,827	192.6
L(3,14)	n	primal	810,725	405,362	1,515.1
L(3,14)	n	dual	361,223	180,611	1,232.9
L(3,14)	n	guided	389,889	194,944	838.1
L(3,18)	y	primal	24,856	12,401	57.2
L(3,18)	y	dual	4,328	2,149	17.4
L(3,18)	y	guided	5,873	2,921	16.2
L(3,19)	y	primal	101,705	50,838	235.6
L(3,19)	y	dual	3,760	1,864	15.6
L(3,19)	y	guided	5,627	2,799	16.0

of constraints that each variable is involved in. While such heuristics are cheap to employ, they are often neither sufficiently aware of the semantics of the constraints nor the problem's structure. However, a number of approaches proposed recently that try to actively modify the search in an online fashion are notable exceptions [5, 16]. Our work is complementary to such approaches.

There has been work on using dual viewpoint models to improve the efficiency with which a CSP can be solved [7, 11]. Given an arbitrarily formulated primal model, a dual viewpoint can be formulated by regarding the variables in the primal as domain values in the dual model and the primal domain values as dual variables. The constraints in the primal can be reformulated in order to be applied in the dual. Channelling constraints between the models ensure that inferences made in one viewpoint can be propagated in the other viewpoint, thus potentially giving efficiency benefits. The dual-viewpoint approach is closely related to earlier work on heuristics that consider both variable and value selection simultaneously [9]. Our work can also be seen as complementary to such approaches.

Recent work on global constraints has envisaged using the information they compute to guide search heuristics. For example, counting the number of solutions using global constraints could be used to inform search heuristics [14]. It has been shown how shortest path information can be used as a value ordering [8]. Others have used a modular search heuristic that exploits global constraints to use heuristic information to inform variable and value choice [4]. These approaches are specific instances of the framework we propose here.

6 CONCLUSIONS

In this paper we have shown that constraints can be used to guide the search process by proposing the next variable and/or value to be branched on. Experimental results on a variety of problem classes show very positive results. There are a number of possible directions

for future work. We would like to develop guiding schemes for other constraints, particularly more global constraints. We will also evaluate the impact of using other measures of pruning on the quality of the guidance, while considering the overhead costs associated with more elaborate schemes.

We regard this work as a step towards a novel type of self-configuring constraint solver. Our vision is one where the search strategy, for problem models developed by non-expert humans, is determined automatically by the guidance provided by individual constraints. Such a vision moves us closer to the goal of declarative, efficient, but most importantly, user-friendly constraint programming.

ACKNOWLEDGEMENTS

This material is based on work supported by Science Foundation Ireland under Grant 00/PI.1/C075. We thank Brahim Hnich for his helpful comments.

REFERENCES

- [1] N. Beldiceanu, 'Global constraints as graph properties on structured networks of elementary constraints of the same type', Technical Report T2000-01, SICS, (January 2000).
- [2] C. Bessiere and J.-C. Regin, 'MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems', in *Proceedings of CP-1996*, LNCS 1118, pp. 61–75, (1996).
- [3] C. Bessiere, B. Zanuttini, and C. Fernandez, 'Measuring search trees', in *Proceedings ECAI'04 workshop on Modelling and Solving Problems with Constraints*, pp. 31–40, (2004).
- [4] S. Bourdais, P. Galinier, and G. Pesant, 'Hibiscus: A constraint programming application to staff scheduling in health care.', in *Proceedings of CP-2003*, LNCS 2833, pp. 153–167, (2003).
- [5] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, 'Boosting systematic search by weighting constraints', in *Proceedings of ECAI-2004*, pp. 146–150, (2004).
- [6] D. Brélaz, 'New methods to color the vertices of a graph', *Communications of the ACM*, **22**(4), 251–256, (1979).
- [7] B. M. W. Cheng, Kenneth M. F. Choi, Jimmy Ho-Man Lee, and J. C. K. Wu, 'Increasing constraint propagation by redundant modeling: an experience report', *Constraints*, **4**(2), 167–192, (1999).
- [8] S. Demassey, G. Pesant, and L.-M. Rousseau, 'Constraint programming based column generation for employee timetabling.', in *CPAIOR*, pp. 140–154, (2005).
- [9] P. A. Geelen, 'Dual viewpoint heuristics for binary constraint satisfaction problems.', in *ECAI*, pp. 31–35, (1992).
- [10] R.M. Haralick and G.L. Elliott, 'Increasing tree search efficiency for constraint satisfaction problems', *Artif. Intell.*, **14**(4), 263–313, (1980).
- [11] B. Hnich, B.M. Smith, and T. Walsh, 'Dual modelling of permutation and injection problems.', *J. Artif. Intell. Res.*, **21**, 357–391, (2004).
- [12] K. Kuchcinski, 'Constraints-driven scheduling and resource assignment', *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, **8**(3), 355–383, (July 2003).
- [13] D.G. Mitchell, 'Resolution and constraint satisfaction.', in *Proceedings of CP-2003*, LNCS 2833, pp. 555–569, (2003).
- [14] G. Pesant, 'Counting solutions to CSPs: A structural approach', in *Proceedings of IJCAI-2005*, pp. 260–265, (2005).
- [15] J.-F. Puget. The next challenge for CP: Ease of use. Invited Talk at CP-2004.
- [16] P. Refalo, 'Impact-based search strategies for constraint programming', in *Proc. of Principles and Practice of Constraint Programming (CP)*, volume 3258, pp. 557–571, (2004).
- [17] J.-C. Regin, 'A filtering algorithm for constraints of difference in CSPs', in *Proc. of AAAI*, volume 1, pp. 362–367, (1994).
- [18] B.M. Smith and P. Sturdy, 'Value ordering for finding all solutions.', in *IJCAI*, pp. 311–316, (2005).
- [19] M. Wallace, 'Practical applications of constraint programming', *Constraints*, **1**(1–2), 139–168, (1996).

Beyond Singleton Arc Consistency

M.R.C. van Dongen¹

Abstract. *Shaving algorithms*, like *singleton arc consistency* (SAC), are currently receiving much interest. They remove values which are not part of any solution. This paper proposes an efficient shaving algorithm for enforcing stronger forms of consistency than SAC. The algorithm is based on the notion of weak *k-singleton arc consistency*, which is equal to SAC if $k = 1$ but stronger if $k > 1$. This paper defines the notion, explains why it is useful, and presents an algorithm for enforcing it. The algorithm generalises Lecoutre and Cardon's algorithm for establishing SAC. Used as pre-processor for MAC it improves the solution time for structured problems. When run standalone for $k > 1$, it frequently removes more values than SAC at a reasonable time. Our experimental results indicate that at the SAC phase transition, it removes many more values than SAC-1 for $k = 16$ in less time. For many problems from the literature the algorithm discovers *lucky solutions*. Frequently, it returns satisfiable CSPs which it proves *inverse consistent* if all values participate in a lucky solution.

1 Introduction

The notion of *local consistency* plays an important role in constraint satisfaction, and many such notions have been proposed so far. For the purpose of this paper we restrict our attention to binary Constraint Satisfaction Problems (CSPs).

A CSP having variables X is (s, t) -consistent [6] if any consistent assignment to the s variables in S can be extended to some consistent assignment to the $s + t$ variables in $S \cup T$, for any sets $S \subseteq X$ and $T \subseteq X \setminus S$ such that $|S| = s$ and $|T| = t$. Local consistency notions are usually enforced by removing tuples and/or recording nogoods. As s increases enforcing (s, t) -consistency becomes difficult because it requires identifying and recording $\mathcal{O}(\binom{n}{s}d^s) = \mathcal{O}(n^s d^s)$ nogoods, where n is the number of variables in the CSP and d is the maximum domain size. For example, $(2, 1)$ -consistency, also known as path consistency [9], is in the most benign class beyond $s = 1$ but it is considered prohibitive in terms of space requirements. The space complexity issues arising with increasing s are the reason why practical local consistency algorithms keep s low. Usually, this means setting s to 1, which means enforcing consistency by removing values from the domains.

Shaving algorithms also enforce consistency by removing values from the domains. They fix variable-value assignments and remove values that inference deems inconsistent. They terminate if no values can be removed. A shaving algorithm, which is currently receiving much attention, is singleton arc consistency (SAC) [4; 5; 11; 1; 8].

¹ Cork Constraint Computation Centre (4C). 4C is supported by Science Foundation Ireland under Grant 00/PI.1/C075.

Another computational complexity source of (s, t) -consistency is the requirement that each consistent assignment to the s variables in S be extendable to a consistent assignment to the $s+t$ variables in $S \cup T$, for each set $S \subseteq X$ having Cardinality s and any set $T \subseteq X \setminus S$ having Cardinality t . Relaxing this requirement by substituting *some* for *any* would make it considerably more easy to enforce the resulting consistency notion, which we call *weak (s, t) -consistency*.

At first glance weak (s, t) -consistency may seem too weak to do useful propagation. However, it has strong advantages:

1. We don't have to find a consistent extending assignment to the variables of *all* sets T of size t . This is especially useful if the problem is already (s, t) -consistent or if s is small.
2. By cleverly selecting T we may still find inconsistencies. For example, if there is no consistent assignment to the variables in $S \cup T$ for the first set T then the current assignment to the variables in S cannot be extended.
3. By increasing t we can enforce levels of consistency which, in a certain sense, are stronger and stronger.

This paper proposes to exploit these strengths, proposing an algorithm which switches between enforcing weak and full consistency, taking the best from both worlds. Given a consistent assignment to the variables in S , the algorithms only seek a consistent assignment to $S \cup T$ for a, cleverly chosen, *first* set T , for which such consistent assignment is unlikely to exist. Should there be a consistent assignment then the current assignment to S is weakly consistent and otherwise it is inconsistent. If $s = 1$ then this allows us to prune the value that is currently assigned to the variable in S .

From now on let $s = 1$. We apply the idea of switching between weak and full consistency to *k-singleton arc consistency*, which generalises SAC [4; 5; 11; 1; 8]. Here, a CSP, \mathcal{P} , is *k-singleton arc consistent* (*weakly k-singleton arc consistent*) if for each variable x , and each value v in its domain, the assignment $x := v$ can be extended by assigning values to each (some) selection of $k - 1$ other variables such that \mathcal{P} can be made arc consistent. SAC is equivalent to *k-singleton* and weak *k-singleton arc consistency* if $k = 1$ but weaker if $k > 1$. Switching between weak and full *k-singleton arc consistency* allows us, in a reasonable time, to enforce stronger levels of consistency, which go beyond SAC. Using an algorithm which enforces weak *k-singleton arc consistency* as a pre-processor for MAC [12] allows the solution of CSPs which cannot be solved by other known algorithms in a reasonable amount of time. Our algorithm is inspired by Lecoutre and Cardon's algorithm for enforcing SAC [8]. Like theirs it frequently detects *lucky solutions* [8] (solutions discovered while enforcing consistency), making search unnecessary for certain

satisfiable problems. Going beyond SAC, our algorithm detects certain *unsatisfiable* problems without search, including problems that *can* be made SAC. A problem is *inverse consistent* if all values participate in some solution. Our algorithms make and prove many problems inverse consistent, including (un)modified radio link frequency assignment problems, and forced random binary problems. Sometimes this is done more quickly than it takes SAC-1 to make these problems SAC.

We start by recalling definitions of constraint satisfaction, by recalling existing notions of consistency, and by introducing new consistency notions. This includes *weak k-singleton arc consistency*. Next we describe an algorithm for enforcing it. Finally, we present experimental results and conclusions.

2 Constraint Satisfaction

A *binary constraint satisfaction problem* (CSP), \mathcal{P} , comprises a set of n variables X , a finite domain $D(x)$ for each $x \in X$, and a set of e binary constraints. The maximum domain size is d . Each constraint is a pair (σ, ρ) , where $\sigma = \langle x, y \rangle \in X^2$ is the *scope*, and $\rho \subseteq D(x) \times D(y)$ is the *relation* of the constraint. Without loss of generality we assume that $x \neq y$ for any scope $\langle x, y \rangle$. We write $\mathcal{P} \neq \perp$ if \mathcal{P} has no empty domains.

Let $\langle \langle x, y \rangle, \rho \rangle$ be a constraint. Then $v \in D(x)$ and $w \in D(y)$ are *arc consistent* if $\{v\} \times D(y) \cap \rho \neq \emptyset$ and $D(x) \times \{w\} \cap \rho \neq \emptyset$. The *arc consistent equivalent* of \mathcal{P} , denoted $ac(\mathcal{P})$, is obtained from \mathcal{P} by repeatedly removing all arc inconsistent values (and, if needed, adjusting constraint relations).

The *density* of \mathcal{P} is defined $2e/(n^2 - n)$. The *tightness* of constraint $\langle \langle x, y \rangle, \rho \rangle$ is defined $|\rho|/|D(x) \times D(y)|$.

An *assignment* is a partial function with domain X . A *k-assignment* assigns values to k variables (only). By abuse of notation we write $\{x_1 = f(x_1), \dots, x_k = f(x_k)\}$ for k -assignment f . Let $f = \{x_1 = v_1, \dots, x_k = v_k\}$ be a k -assignment. We call f *consistent* if $k = 1$ and $v_1 \in D(x_1)$ or $k > 1$ and $\langle v_i, v_j \rangle \in \rho$ for each constraint $\langle \langle x_i, x_j \rangle, \rho \rangle$ such that $1 \leq i, j \leq k$. A consistent n -assignment is a *solution*. $\mathcal{P}|_f$ is obtained from \mathcal{P} by substituting $D(x_i) \cap \{v_i\}$ for $D(x_i)$ for all i such that $1 \leq i \leq k$.

3 Consistency

Definition 1 ((s, t)-consistency). A CSP with variables X is (s, t) -consistent if for any variables $S = \{x_1, \dots, x_s\} \subseteq X$ and any variables $\{x_{s+1}, \dots, x_{s+t}\} \subseteq X \setminus S$, any consistent s -assignment to x_1, \dots, x_s is extendable to some consistent $(s+t)$ -assignment to x_1, \dots, x_{s+t} .

Enforcing (s, t) -consistency may require processing (almost) all assignments to all combinations of s variables and all assignments to all combinations of t additional variables. As s and t become large, enforcing (s, t) -consistency usually results in a large average running time and (generally) requires $\mathcal{O}(n^s d^t)$ space for recording nogoods. The following relaxes (s, t) -consistency by substituting *some* for the second occurrence of *any* in the definition of (s, t) -consistency.

Definition 2 (Weak (s, t) -Consistency). A CSP with variables X is weakly (s, t) -consistent if for any variables $S = \{x_1, \dots, x_s\} \subseteq X$ and some variables $\{x_{s+1}, \dots, x_{s+t}\} \subseteq X \setminus S$, any consistent s -assignment to x_1, \dots, x_s is extendable to some consistent $(s+t)$ -assignment to x_1, \dots, x_{s+t} .

A CSP is *inverse k -consistent* [6] if it is $(1, k-1)$ -consistent. Inverse consistency does not require additional constraints and can be enforced by shaving. A CSP is *inverse consistent* if it is inverse n -consistent. It is *weakly inverse k -consistent* if it is weakly $(1, k-1)$ -consistent. Then (weak) inverse K -consistency implies (weak) inverse k -consistency if $K \geq k$.

A CSP is called *k-consistent* if it is $(k-1, 1)$ -consistent and it is called *arc consistent* if it is 2-consistent. The following formally defines singleton arc consistency.

Definition 3 (Singleton Arc Consistency). A CSP, \mathcal{P} , with variables X is called *singleton arc consistent* (SAC) if

$$(\forall x_1 \in X)(\forall v_1 \in D(x_1))(ac(\mathcal{P}|_{\{x_1=v_1\}}) \neq \perp).$$

The following seems a natural generalisation of SAC.

Definition 4 (k -Singleton Arc Consistency). A CSP \mathcal{P} with variables X is called *k-singleton arc consistent* if

$$\begin{aligned} & (\forall x_1 \in X)(\forall v_1 \in D(x_1))(\forall \{x_2, \dots, x_k\} \subseteq X \setminus \{x_1\}) \\ & (\exists \{v_2, \dots, v_k\} \in D(x_2) \times \dots \times D(x_k)) \\ & (ac(\mathcal{P}|_{\{x_1=v_1, \dots, x_k=v_k\}}) \neq \perp). \end{aligned}$$

We define *weak k-singleton arc consistency* (weak k -SAC) by substituting an existential quantifier for the last universal quantifier in Definition 4. Then weak 1-SAC is equivalent to 1-SAC and SAC, and (weak) K -SAC implies (weak) inverse k -consistency and (weak) k -SAC if $K \geq k$.

4 A Weak k -SAC Algorithm

This section presents our algorithms, which use greedy search to establish a weakly k -SAC equivalent of the input CSP \mathcal{P} . They exploit that $ac(\mathcal{P}|_{\{x_1=v_1, \dots, x_l=v_l\}}) \neq \perp$ implies $ac(\mathcal{P}|_{\{x_{i_1}=v_{i_1}, \dots, x_{i_k}=v_{i_k}\}}) \neq \perp$, for any $\{x_{i_1}, \dots, x_{i_k}\} \subseteq \{x_1, \dots, x_l\}$. This generalises the SAC algorithms in [8], which use greedy search, exploiting that $ac(\mathcal{P}|_{\{x_1=v_1, \dots, x_l=v_l\}}) \neq \perp$ implies $ac(\mathcal{P}|_{\{x_{i_1}=v_{i_1}\}}) \neq \perp$, for any $\{x_{i_1}\} \subseteq \{x_1, \dots, x_l\}$.

The algorithms are depicted in Figure 1. The outer **while** loop of *wksac* is executed while there are changes, while there is no inconsistency, and while no lucky solution has been found. (Removing the statement *solved := true* in *extendable* prohibits finding lucky solutions.) The second outer **while** loop selects the next variable, x . The inner-most **while** loop removes singleton inconsistent values. For any remaining value, v , *wksac* tries to extend the assignment $x = v$, to some K -SAC assignment, for some $K \geq k$ by executing *extendable*($k-1, wksac, solved, X \setminus \{x\}$). If this fails then v is removed. The underlying arc consistency algorithm is *ac*.

Like SAC3 and SAC3+ [8] *extendable* also searches for assignments of length greater than k . This allows the discovery of *lucky solutions* [8] as part of consistency processing. If it finds a k -SAC assignment then *extendable* allows no digressions when trying to find an extending K -SAC assignment, for $K > k$. This can be generalised to more digressions.

The space complexity of *wksac* is equal to the space complexity of MAC plus the space required for storing the array *wksac*[\cdot, \cdot]. The space complexity of *wksac*[\cdot, \cdot] is $\mathcal{O}(nd)$, which cannot exceed the space complexity of MAC. Therefore, the space complexity of *wksac* is equal to the space complexity of MAC. The outer loop of *wksac* is executed $\mathcal{O}(nd)$ times. For

each of the $\mathcal{O}(nd)$ values, finding an extending k -SAC assignment takes $\mathcal{O}(d^{k-1}T)$ time, where T is the time complexity of ac . For each k -SAC assignment it takes $\mathcal{O}((n-k)T)$ time for trying to find a lucky solution. Therefore, $wksac$'s time complexity is $\mathcal{O}(n^2d^{k+1}T + (n-k)n^2d^2T)$.

Termination and correctness proofs are straightforward. The following two propositions provide the basis for a correctness proof. Proofs are omitted due to space restrictions.

Proposition 1. *If $extendable(k-1, wksac, solved, X \setminus \{x\})$ succeeds then the assignment $x = v$ extends to a K -SAC assignment, for some $K \geq k$. Otherwise $x = v$ is inconsistent.*

Proposition 2. *If $wksac(k, X)$ succeeds then it computes a solution or some weakly k -SAC equivalent of the input CSP. If it fails then the input CSP is unsatisfiable.*

```

function wksac(in k, in X) do
  local variables consistent, change, solved;
  consistent := ac(); change := true; solved := false;
  while consistent and change and ~solved do
    foreach value v in the domain of each variable x do
      wksac[x, v] := false;
    od;
    change := false; vars := X;
    while consistent and ~change and ~solved and vars ≠ ∅ do
      Select and remove any variable x from vars;
      vals := { v ∈ D(x) : ~wksac[x, v] };
      while vals ≠ ∅ and consistent and ~change do
        Select and remove any v from vals;
        assign v to x;
        if ac() and extendable(k-1, wksac, solved, X \ {x})
          undo ac(); unassign v; wksac[x, v] := true;
        else
          remove v from D(x);
          change := true; undo ac(); unassign v; consistent := ac();
        fi;
      od;
    od;
    return consistent;
  od;

function extendable(in k, in/out wksac, in/out solved, in X) do
  local variables consistent, extendable, values;
  if X = ∅
    solved := true; /* Remove line if not used for solving. */
    return true;
  else
    select any x from X;
    values := D(x);
    extendable := false;
    while ~extendable and values ≠ ∅ do
      select and remove any v from values;
      assign v to x;
      if ac() and extendable(k-1, wksac, solved, X \ {x}) then
        wksac[x, v] := true; extendable := true;
      fi;
      undo ac(); unassign v;
    od;
    return extendable or k ≤ 0;
  fi;
od;

```

Figure 1. Algorithms for enforcing weak k -SAC.

The algorithms in Figure 1 do not represent efficient implementations. The following are suggestions for improvement.

The computation time usually improves if $extendable$ first selects variables from the set $vars$ used by $wksac$, and if it first selects $v \in D(x)$ such that $wksac[x, v] = false$.

Finally, note that a good variable ordering is essential. For example, given a CSP consisting of k connected components, it makes no sense to select one variable from each component since *any* assignment to these variables is consistent, not allowing any pruning. The conflict directed variable order dom/w_{deg} [3] is good for our algorithms. This heuristic uses the current degrees and the numbers of previously failed revisions, selecting a variable that quickly leads to a dead-end [3].

Remember that $k = 1$. If $extendable$ has not been called yet then a dead-end immediately proves the assignment $x = v$ inconsistent. Otherwise, it immediately proves $x = v$ weakly K -SAC, for some $K \geq k$. Ties are broken lexically. Values for x are selected preferring value v such that $wksac[x, v] = false$, breaking ties using the order $svoh_2$ [10]. Other variable and value orders usually result in worse results.

For certain CSPs and $K > k > 0$, $wksac$ may prune *more* for k than for K . However, experimental evaluation of the algorithm indicates that this usually is not the case.

5 Experimental Results

To investigate their behaviour, we experimentally compared SAC-1, and $wksac$ for $k \in \{1, 2, 4, 8, 16\}$. We (1) investigate their shaving capabilities for problems known from the literature, (2) investigate their behaviour for random problems, and (3) investigate their use as a preprocessor for MAC [12].

All algorithms were implemented in C. They were run on a Linux 2.8GHz PC, having 500Mb of RAM. The underlying arc consistency algorithm is AC-3. Some authors prefer optimal arc consistency algorithms, but we feel that AC-3 is a good general purpose algorithm. For example, the best solvers in the binary and overall categories of the First International CSP Solver Competition are based on AC-3 [13].

Shaving Problems from the Literature The algorithms have been applied to known problems, including forced random binary problems `frb`, RLFAP, modified RLFAP, attacking prime queen problems `qa1`, and job-shop instances `enddr1-10-by-5-10` and `enddr2-10-by-5-2` (they are called `js-1` and `js-2` in [8]). All problems are described in [2] and may be downloaded from <http://cpai.ucc.ie/>.

Table 1 lists the main results for the first experiment. It lists the problem instances, and for each instance: the number of variables, the number of values, the number of removed values, and the time this took for SAC-1 and $wksac$ for $k \in \{1, 2, 4, 8, 16\}$. The satisfiable instances are indicated by a + in the column `sat`. An i in the column `del` indicates that the CSP has been made and proved inverse consistent. The column `min` (`max`) lists the minimum (maximum) cardinalities of sets of lucky solutions, which were found in the process of enforcing weak 1-SAC. These sets are found when there are no more constraints among the future variables, in which case the number of solutions is equal to the product of the domain sizes. The column `count` lists the total number of such solutions. Due to the number of sets of lucky solutions and their size, it was impossible, for some problem classes, to compute the actual number of lucky solutions, in which case the entry for column `count` has a question mark. [8] also report the finding of lucky solutions. [8] report 0 lucky solutions for `qa-5`, `qa-6`, 1 for `scen5` and `enddr2-10-by-5-2`, 4 for `enddr1-10-by-5-10`, 10 for `graph14`, and 16 for `scen2`, which is less than the numbers reported here. Perhaps these differences are caused by differences in variable and value ordering.

Many values in the job-shop problems are inverse consistent. At the First International CSP Solver Competition they were difficult to solve by MAC [13]. This may explain why it is difficult to enforce higher level of k -SAC using search: for $k \in \{8, 16\}$ it takes too much time. For the attacking prime

Table 1. Results of shaving problems from the literature using SAC-1 and *wksac* for different values of k

Instance	sat	vars	vals	SAC-1			lucky		$k = 1$ del	$k = 1$ time	$k = 2$ del	$k = 2$ time	$k = 4$ del	$k = 4$ time	$k = 8$ del	$k = 8$ time	$k = 16$ del	$k = 16$ time
				del	time	count	min	max										
frb30-15-1	+	30	450	0	0.04	0	0	0	0	0.10	0	0.10	0	0.09	i 410	6.55	i 410	0.54
frb30-15-2	+	30	450	0	0.04	0	0	0	0	0.10	0	0.09	0	0.10	i 413	7.02	i 413	1.16
frb35-17-1	+	35	595	0	0.05	0	0	0	0	0.14	0	0.14	0	0.14	0	1.72	i 559	3.19
frb35-17-2	+	35	595	0	0.06	0	0	0	0	0.14	0	0.14	0	0.14	0	0.90	i 552	7.54
frb40-19-1	+	40	760	0	0.08	0	0	0	0	0.21	0	0.21	0	0.21	0	0.51	i 701	15.05
frb40-19-2	+	40	760	0	0.07	0	0	0	0	0.22	0	0.21	0	0.21	0	0.52	i 717	28.18
scen-1	+	916	36200	0	20.90	42	1	1	0	6.46	0	6.47	0	6.47	0	6.48	0	6.48
scen-2	+	200	8004	0	4.89	56	1	1	1	0	1.28	i 0	1.28	i 0	1.29	i 0	1.28	
scen-5	+	400	15768	13814	0.97	46	1	1	i 13814	0.28	76	4803.89						
scen-11	+	680	26856	0	15.17	11	1	1	0	4.88	0	4.89	0	4.88	0	4.78	76	4803.89
scen1-f8	+	916	29496	6704	6.29	17	1	1	6704	2.55	6704	2.55	6704	2.56	6704	2.55	6704	2.58
scen2-f24	+	200	4025	0	0.57	25	1	1	0	0.27	0	0.27	0	0.27	0	0.32	0	0.42
scen3-f10	+	400	12174	3726	2.20	22	1	1	3726	1.43	3726	1.43	3726	1.56	3728	1.54	3746	1.78
scen6-w1	+	200	8020	1580	2.78	58	1	1	1580	0.78	1580	0.77	1616	0.76	1616	0.76	1616	0.80
scen7-w1-f4	+	400	14563	6286	1.88	62	1	1	6286	1.10	6286	1.10	6318	0.90	6318	0.89	6326	1.15
scen1-f9	-	916	28594	7628	5.24	0	0	0	7628	2.47	7628	2.47	7640	4.75	28596	2.46	28596	2.89
scen2-f25	-	200	3918	106	0.58	0	0	0	106	0.35	106	0.36	110	0.40	3918	0.54	3918	1.50
scen3-f11	-	400	11963	3934	1.59	0	0	0	3934	1.79	3934	1.79	3980	1.85	11966	1.36	11966	0.41
scen6-w1-f22	-	200	7716	2082	2.34	0	0	0	2082	1.25	2082	1.25	7716	0.40	7716	0.18	7716	0.22
scen6-w1-f3	-	200	7718	2082	2.34	0	0	0	2082	1.25	2082	1.25	7718	0.40	7718	0.18	7718	0.22
scen11-f1	-	680	26524	332	13.94	0	0	0	2474	0.59	2474	0.59	7518	0.07	7518	0.06	7518	0.09
qa-5	+	26	531	9	0.16	0	0	0	9	0.10	9	0.10	12	0.45	i 386	24.06		
qa-6	+	37	1302	48	2.19	0	0	0	48	0.78	48	0.80	48	0.81	67	1.86	127	1923.01
qa-10	+	101	10015	373	311.92	0	0	0	373	93.91	373	94.64	393	440.62	416	306.66	—	—
enddr1-10-by-5-10	+	50	5760	0	16.07	?	532	15.3e6	0	5.85	0	5.83	0	5.83	—	—	—	—
enddr2-10-by-5-2	+	50	6315	0	25.95	?	21	50.5e3	0	10.68	0	10.70	0	10.70	—	—	—	—

queen problems *wksac* is removing more values as k increases, but for $k = 16$ it needs too much time.

For these structured problems the new algorithms are about as efficient as SAC-1 for $k = 1$, if not better. All unsatisfiable modified *rlfap* instances (the unsatisfiable unmodified instances are proved inconsistent by arc consistency) are proved inconsistent by *wksac* for $k \geq 8$, whereas SAC-1 fails to prove some of these instances inconsistent. Note that instance **scen11-f1** is very difficult and, to the best of our knowledge, has not been classified; days of MAC search could not solve it. However, enforcing weak 8-SAC proves it inconsistent within an hour. Finally, some satisfiable problems (they are not all listed in Table 1) are made and proved inverse consistent within a few seconds. Here a problem is proved inverse consistent if all values participate to some lucky solution. For example, **scen2** is proved to be already inverse consistent, **graph2**, **scen3**, **scen4**, and **scen5** are made and proved inverse consistent by making them weak k -SAC ($k \geq 1$), and all fifteen instances from the classes **frb-30-15**, **frb-35-17**, and **frb-40-19** are made and proved inverse consistent by making them weakly k -SAC for $k = 8$ or $k = 16$.

Shaving Random Problems We now study the behaviour of the algorithms for random model B problems [7]. A model B class is typically denoted $\langle n, d, D, T \rangle$, where n is the number of variables, d is the uniform domain size, D is the density, and T is the uniform tightness. Results are presented for the same class of problems as presented in [1; 8]. For each tightness $t = 0.05 i$, $2 \leq i \leq 18$, the average shaving time is over 50 random instances from $\langle 100, 20, 0.05, t \rangle$.

Figure 3 does not depict all data. For $T \leq 0.50$ all algorithms remove less than 0.06, and for $T \geq 0.8$ they remove about 1989.6 values on average. Figure 3 confirms that SAC and weak 1-SAC are equivalent. Comparing SAC-1 and **w1sac** in Figure 2, **w1sac** is better in time when T is small and large. However, near the SAC complexity peak SAC-1 is about three times quicker than **w1sac**. The majority of the problems in that region are unsatisfiable and most values are SAC. Typically, SAC-1 will find that a value is SAC and stop. The extra work put in by **w1sac** in trying to find a lucky solution will fail at a shallow level. This work cannot do pruning and does not lead to many values that were not known to be SAC.

The weak k -SAC algorithms only behave differently near the SAC phase transition. The higher k the more values are removed. At $T = 0.7$, the nearest point to the SAC phase tran-

sition, **weak16sac** outperforms SAC-1 marginally in time and significantly in the number of removed values. At $T = 0.65$ **w16sac** removes about 114.2 values on average, whereas all other algorithms remove between 2 and 3 values on average. However, **w16sac** spends (much) more time. Clearly the algorithms cannot be compared in time at $T = 0.65$.

Search We now compare the algorithms as a preprocessor for MAC. The first solver is MAC, denoted **mac**, the second is **sac+mac**, which is SAC-1 followed by MAC, and the third and fourth solvers are **w1sac+mac** and **w8sac+mac**, i.e. *wksac* for enforcing weak k -singleton arc consistency followed by MAC, for $k \in \{1, 8\}$. All solvers used the variable ordering *dom/w_{deg}* [3] and the value ordering *svoh₂* [10], breaking ties lexicically. The support counters [10] for the value ordering are initialised after establishing initial arc consistency. Should *wksac* prune more values, then they are also initialised before search.

Table 2 lists the results. The column **sat** denotes the satisfiability of the instances. A + and an L indicates satisfiable instances, the L indicating the discovery of lucky solutions. For all problems, if lucky solutions are found by **w1sac+mac** then they are also found by **w8sac+mac** and vice versa.

Overall, and these results are typical for these problems, **sac+mac** performs worse in time and checks than **wksac+mac**. **Mac** performs better than **sac+mac** for some instances, especially some **graph** instances, otherwise the two algorithms are about equally efficient. Compared to **w1sac+mac** and **w8sac+mac** it is clear that **sac+mac** is worse in time and checks. Compared to **mac** the results are not so clear but overall **w1sac+mac** and **w8sac+mac** are better. The only exceptions are for “easy” problems, for which **mac** is easy. For these problems there is no need to establish more consistency before search and this results in slightly more solution time. We have also observed this for unsatisfiable problem instances where **sac** alone is sufficient to detect the inconsistency. For example, for the queens-knight problems **qk1** [2] SAC-1 is more efficient than the weak SAC algorithms. However, these problems are relatively easy and do not require much time, even with *wksac*. For the problems that are difficult for **mac** and **sac+mac** the two need a considerable amount of time more than *wksac*.

It is recalled that it turned out impossible to compute the weak k -SAC equivalent of the job-shop instances for $k = 8$ and $k = 16$. However, when the algorithms are used to find solutions, they perform much better and find lucky solutions. Lucky solutions are also found for all satisfiable **rlfap**, and all

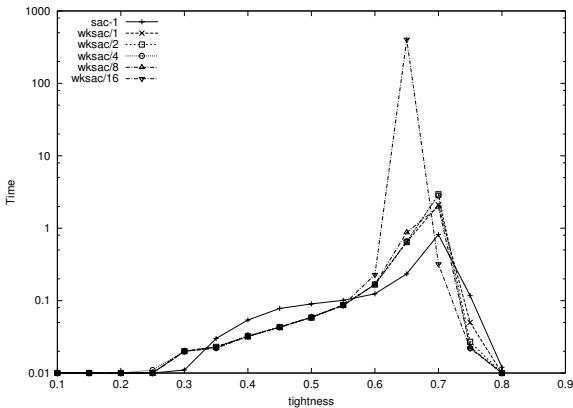


Figure 2. Shaving time for random B class $\langle 100, 20, 0.05, t \rangle$ for different algorithms, where $0.1 \leq T \leq 0.9$

satisfiable modified **r1fap** instances, including instances from these classes for which no results are presented in Table 1. It is interesting to note that if lucky solutions are found, it takes the same amount of checks for $k = 8$ and $k = 16$. This may indicate that both algorithms carry out exactly the same amount of work, which seems possible only if, in addition, they make the same decisions about value and variable ordering. If this is true, then it is probably because these problems are loose, making any arc consistent 1-assignment, easily extendable to an arc consistent k -assignment for $k \geq 8$, which makes it impossible to prune more for **w8sac** than for **w1sac**.

Table 2. Problem solving capabilities of search algorithms.

Instance	mac		sac+mac		w1sac+mac		w8sac+mac	
	time	checks	time	checks	time	checks	time	checks
scen1	L	0.15	2.525e6	3.62	78.121e6	0.16	2.511e6	0.16
scen2	L	0.09	0.564e6	0.92	8.199e6	0.10	0.595e6	0.10
scen5	L	0.61	9.296e6	11.49	246.425e6	0.53	8.545e6	0.53
scen11	L	0.39	4.426e6	5.49	10.210e6	0.10	4.987e6	0.10
scen1-f8	L	0.08	0.723e6	0.49	10.923e6	0.05	0.961e6	0.05
scen2-f24	L	0.04	0.573e6	2.24	33.830e6	0.04	0.722e6	0.04
scen6-w1	L	0.07	8.857e6	1.54	22.930e6	0.08	9.995e6	0.08
scen1-f4	L	0.84	8.857e6	4.56	70.602e6	2.51	25.310e6	2.29
scen2-f25	-	1.44	15.261e6	1.91	25.102e6	0.43	5.667e6	0.54
scen3-f11	-	0.88	8.046e6	2.16	32.568e6	1.94	20.443e6	1.34
scen6-w1-f2	-	2.09	25.712e6	2.08	28.949e6	1.11	13.898e6	0.15
scen6-w1-f3	-	0.84	11.878e6	1.37	19.030e6	0.53	6.631e6	0.06
scen6-w2	-	0.36	4.382e6	0.35	4.290e6	0.15	1.768e6	0.16
scen7-w1-f5	-	0.11	1.235e6	0.08	1.032e6	0.05	0.731e6	0.06
scen9-w1-f3	-	0.16	1.972e6	0.11	1.432e6	0.13	1.654e6	0.13
scen10-w1-f3	-	0.16	1.972e6	0.11	1.432e6	0.13	1.654e6	0.13
graph3	L	0.17	2.185e6	60.11	764.413e6	0.17	2.180e6	0.17
graph10	L	0.72	8.501e6	368.56	198.230e6	0.72	4.891e6	0.72
graph14	L	0.53	9.286e6	15.11	351.031e6	0.53	9.230e6	0.54
qa-5	+	0.29	2.392e6	0.41	5.501e6	0.42	3.652e6	0.06
qa-6	+	106.65	783.722e6	125.22	949.601e6	0.81	12.807e6	1.92
enddr1-10-by-5-10	L	879.89	546.428e6	891.26	974.780e6	0.13	4.405e6	0.13
enddr2-10-by-5-2	L	117.47	3925.611e6	134.53	321.808e6	0.20	7.304e6	0.20

6 Conclusions

This paper introduces *k-singleton consistency* (*k*-SAC) and weak *k*-SAC, which are generalisations of SAC and inverse *k*-consistency. Weak *k*-SAC is equal to SAC if $k = 1$ but stronger if $k > 1$. A weak *k*-SAC algorithm is presented, which uses greedy search. At the SAC phase transition, it removes many more values than SAC-1 for $k = 16$ using less time. Like SAC-3 and SAC-3+ the algorithm sometimes finds lucky solutions. If it does it usually finds many. SAC cannot solve unsatisfiable instances which can be made SAC. However, by enforcing higher degrees of weak SAC some of these instances can be proved inconsistent within a reasonable time. When used as a preprocessor for MAC the algorithm compares favourably to existing algorithms. By increasing the level of weak SAC we are able to solve difficult problems, including **scen11-f1**, which had not been solved before.

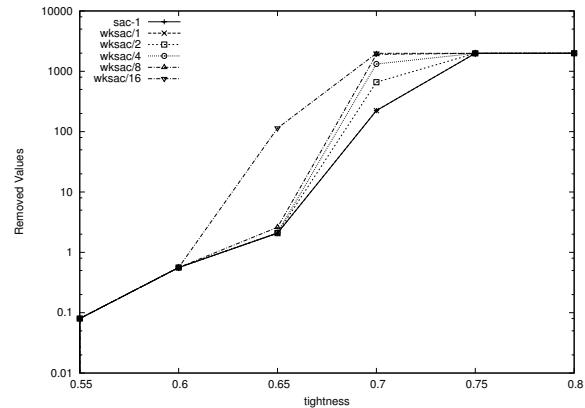


Figure 3. Number of shaved values for random B class $\langle 100, 20, 0.05, t \rangle$ for different algorithms, where $0.5 \leq T \leq 0.8$

REFERENCES

- [1] C. Bessière and R. Debruyne, ‘Optimal and suboptimal singleton arc consistency algorithms’, in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 54–59, (2005).
- [2] F. Boussemart, F. Hemery, and C. Lecoutre, ‘Description and representation of the problems selected for the first international constraint satisfaction problem solver competition’, in *Proceedings of the Second International Workshop on Constraint Propagation And Implementation*, volume II, pp. 7–26, (2005).
- [3] F. Boussemart, F. Hemery, C. Lecoutre, and L. Saïs, ‘Boosting systematic search by weighting constraints.’, in *Proceedings of the Sixteenth European Conference on Artificial Intelligence*, eds., Ramon López de Mántaras and Lorenza Saitta, pp. 146–150, (2004).
- [4] R. Debruyne and C. Bessière, ‘Some practicable filtering techniques for the constraint satisfaction problem’, in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI’97)*, ed., M.E. Pollack, pp. 412–417, (1997).
- [5] R. Debruyne and C. Bessière, ‘Domain filtering consistencies’, *Journal of Artificial Intelligence Research*, **14**, 205–230, (2001).
- [6] E.C. Freuder, ‘A sufficient condition for backtrack-bounded search’, *Journal of the ACM*, **32**(4), 755–761, (1985).
- [7] I.P. Gent, E. MacIntyre, P. Prosser, B. Smith, and T. Walsh, ‘Random constraint satisfaction: Flaws and structure’, *Journal of Constraints*, **6**(4), 345–372, (2001).
- [8] C. Lecoutre and S. Cardon, ‘A greedy approach to establish singleton arc consistency’, in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, (2005).
- [9] A.K. Mackworth, ‘Consistency in networks of relations’, *Artificial Intelligence*, **8**, 99–118, (1977).
- [10] D. Mehta and M.R.C. van Dongen, ‘Static value ordering heuristics for constraint satisfaction problems’, in *Proceedings of the Second International Workshop on Constraint Propagation And Implementation*, ed., M.R.C. van Dongen, pp. 49–62, (2005).
- [11] P. Prosser, K. Stergiou, and T. Walsh, ‘Singleton consistencies’, in *Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming, CP’2000*, ed., R. Dechter, pp. 353–368, (2000).
- [12] D. Sabin and E.C. Freuder, ‘Contradicting conventional wisdom in constraint satisfaction’, in *Proceedings of the Eleventh European Conference on Artificial Intelligence*, ed., A.G. Cohn, pp. 125–129. John Wiley and Sons, (1994).
- [13] *Proceedings of the Second International Workshop on Constraint Propagation And Implementation*, ed., M.R.C. van Dongen, volume II, 2005.

Symmetry Breaking using Value Precedence

Toby Walsh¹

Abstract. We present a comprehensive study of the use of value precedence constraints to break value symmetry. We first give a simple encoding of value precedence into ternary constraints that is both efficient and effective at breaking symmetry. We then extend value precedence to deal with a number of generalizations like wreath value and partial interchangeability. We also show that value precedence is closely related to lexicographical ordering. Finally, we consider the interaction between value precedence and symmetry breaking constraints for variable symmetries.

1 INTRODUCTION

Symmetry is an important aspect of many search problems. Symmetry occurs naturally in many problems (e.g. if we have two identical machines to schedule, or two identical jobs to process). Symmetry can also be introduced when we model a problem (e.g. if we name the elements in a set, we introduce the possibility of permuting their order). We must deal with symmetry or we will waste much time visiting symmetric solutions, as well as parts of the search tree which are symmetric to already visited parts. One simple but highly effective mechanism to deal with symmetry is to add constraints which eliminate symmetric solutions [3].

Two common types of symmetry are variable symmetries (which act just on variables), and value symmetries (which act just on values). With variable symmetries, we have a number of well understood symmetry breaking constraints. For example, many problems can be modelled using matrices of decision variables, in which the rows and columns of the matrix are symmetric and can be freely permuted. We can break such symmetry by lexicographically ordering the rows and columns [4]. Efficient propagators have therefore been developed for such ordering constraints [5, 2].

Value symmetries are also common. However, symmetry breaking for value symmetry is less well understood. In this paper, we study a common type of value symmetry where the values for variables are interchangeable. For example, if we assign orders to machines and two orders are identical, we can swap them in any schedule. We show here how to deal with such symmetry. In particular, we give a simple encoding that breaks all the symmetry introduced by interchangeable values. We also show how this is closely related to the lexicographical ordering constraint.

2 BACKGROUND

A constraint satisfaction problem consists of a set of variables, each with a domain of values, and a set of constraints specifying allowed combinations of values for given subsets of variables. A solution is an assignment of values to variables satisfying the constraints. Finite

domain variables take values which are integers, or tuples of integers taken from some given finite set. Set variables takes values which are sets of integers. A set variable S has a lower bound $lb(S)$ for its definite elements and an upper bound $ub(S)$ for its definite and potential elements.

Constraint solvers typically explore partial assignments enforcing a local consistency property. We consider the two most common local consistencies: arc consistency and bound consistency. Given a constraint C , a *bound support* on C is an assignment of a value to each finite domain variable between its minimum and maximum and of a set to each set variable between its lower and upper bounds which satisfies C . A constraint C is *bound consistent (BC)* iff for each finite domain variable, its minimum and maximum values belong to a bound support, and for each set variable S , the values in $ub(S)$ belong to S in at least one bound support and the values in $lb(S)$ belong to S in all bound supports. Given a constraint C on finite domain variables, a *support* is assignment to each variable of a value in its domain which satisfies C . A constraint C on finite domains variables is *generalized arc consistent (GAC)* iff for each variable, every value in its domain belongs to a support.

A *variable symmetry* is a bijection on variables that preserves solutions. For example, in a model of the rehearsal problem (prob039 in CSPLib) in which we assign musical pieces to time slots, we can always invert any schedule. This is a reflection symmetry on the variables. A *value symmetry* is a bijection on values that preserves solutions. For example, in our model of the rehearsal problem, two pieces requiring the same musicians are interchangeable and can be freely permuted in any solution. Note that some authors call these *global* value symmetries as they act globally on values [12]. Finally, a pair of values are *interchangeable* if we can swap them in any solution.

3 VALUE PRECEDENCE

We can break all symmetry between a pair of interchangeable values using a global value precedence constraint [9].

$$\text{PRECEDENCE}([v_j, v_k], [X_1, \dots, X_n])$$

This holds iff $\min\{i \mid X_i = v_j \vee i = n+1\} < \min\{i \mid X_i = v_k \vee i = n+2\}$. Law and Lee give a specialized algorithm for enforcing GAC on such a global constraint [9]. We show here that this is unnecessary. We can encode the constraint efficiently and effectively into a simple sequence of ternary constraints.

We introduce a sequence of 0/1 variables, where $B_i = 1$ if $X_l = v_j$ for some $l < i$. Value precedence prevents us assigning $X_i = v_k$ unless $B_i = 1$. To ensure this, we post the sequence of ternary constraints, $C(X_i, B_i, B_{i+1})$ for $1 \leq i \leq n$ which hold iff $X_i = v_j$ implies $B_{i+1} = 1$, $X_i \neq v_j$ implies $B_i = B_{i+1}$, and $B_i = 0$ implies $X_i \neq v_k$. We also set $B_1 = 0$. We assume that we can enforce GAC on each individual ternary constraint C using a table constraint

¹ National ICT Australia and University of New South Wales, email: tw@cse.unsw.edu.au

or using primitives like implication and equality constraints. As the constraint graph is Berge-acyclic, enforcing GAC on the ternary constraints achieves GAC on $\text{PRECEDENCE}([v_j, v_k], [X_1, \dots, X_n])$. Since C is functional in its first two arguments and there are $O(n)$ ternary constraints, this takes $O(nd)$ time where d is the maximum domain size of the X_i . This is therefore optimal (as was Law and Lee's specialized algorithm). The incremental behavior is also good. Law and Lee's algorithm maintains three pointers, α , β and γ to save re-traversing the vector. A constraint engine will also perform well incrementally on this encoding provided it ignores constraints once they are entailed. Our experimental results support this claim.

4 MULTIPLE VALUE INTERCHANGEABILITY

Many problems involve *multiple* interchangeable values. For example, in a finite domain model of the social golfer problem (prob010 in CSPLib) in which we assign groups to golfers in each week, all values are interchangeable. To break all such symmetry, Law and Lee [9] propose the global constraint:

$$\text{PRECEDENCE}([v_1, \dots, v_m], [X_1, \dots, X_n])$$

This holds iff $\min\{i \mid X_i = v_i \vee i = n+1\} < \min\{i \mid X_i = v_j \vee i = n+2\}$ for all $1 \leq i < j < m$. To propagate this constraint, Law and Lee suggest decomposing it into pairwise precedence constraints of the form $\text{PRECEDENCE}([v_i, v_j], [X_1, \dots, X_n])$ for all $i < j$ [9]. Law has conjectured (page 77 of [8]), that such a decomposition does not hinder GAC propagation. We prove this is not the case.

Theorem 1 *Enforcing GAC on $\text{PRECEDENCE}([v_1, \dots, v_m], [X_1, \dots, X_n])$ is strictly stronger than enforcing GAC on $\text{PRECEDENCE}([v_i, v_j], [X_1, \dots, X_n])$ for all $1 \leq i < j \leq m$.*

Proof: Clearly it is at least as strong. To show strictness, consider $\text{PRECEDENCE}([1, 2, 3, 4], [X_1, X_2, X_3, X_4])$ with $X_1 \in \{1\}$, $X_2 \in \{1, 2\}$, $X_3 \in \{1, 3\}$, and $X_4 \in \{3, 4\}$. Then enforcing GAC on $\text{PRECEDENCE}([1, 2, 3, 4], [X_1, X_2, X_3, X_4])$ prunes 1 from the domain of X_2 . However, $\text{PRECEDENCE}([i, j], [X_1, X_2, X_3, X_4])$ is GAC for all $1 \leq i < j \leq 4$. \diamond

We propose instead a simple encoding of $\text{PRECEDENCE}([v_1, \dots, v_m], [X_1, \dots, X_n])$ into a sequence of ternary constraints. We introduce $n + 1$ finite domain variables, where Y_i records the greatest index of the values used so far in the precedence order. We then post a sequence of ternary constraints, $D(X_i, Y_i, Y_{i+1})$ for $1 \leq i \leq n$ which hold iff $X_i \neq v_j$ for any $j > Y_i + 1$, $Y_{i+1} = Y_i + 1$ if $X_i = v_{1+Y_i}$ and $Y_{i+1} = Y_i$ otherwise. We set $Y_1 = 0$. Again, we achieve GAC on the global constraint simply by enforcing GAC on the individual ternary constraints. This takes $O(nmd)$ time. By comparison, enforcing GAC on the decomposition into precedence constraints between all pairs of interchangeable values takes $O(nm^2d)$ time.

5 PARTIAL INTERCHANGEABILITY

We may have a partition on the values, and values within each partition are interchangeable. For example, in the model of the rehearsal problem in which we assign musical pieces to time slots, we can partition the musical pieces into those requiring the same musicians. Suppose the values are divided into s equivalence classes, then we can break all symmetry with the global constraint:

$$\text{PRECEDENCE}([v_{1,1}, \dots, v_{1,m_1}], \dots, [v_{s,1}, \dots, v_{s,m_s}], [X_1, \dots, X_n])$$

This holds iff $\min\{i \mid X_i = v_{j,k} \vee i = n+1\} < \min\{i \mid X_i = v_{j,k+1} \vee i = n+2\}$ for all $1 \leq j \leq s$ and $1 \leq k < m_j$. This global constraint can be decomposed into s precedence constraints, one for each equivalence class. However, such decomposition hinders propagation.

Theorem 2 *Enforcing GAC on $\text{PRECEDENCE}([v_{1,1}, \dots, v_{1,m_1}], \dots, [v_{s,1}, \dots, v_{s,m_s}], [X_1, \dots, X_n])$ is strictly stronger than enforcing GAC on $\text{PRECEDENCE}([v_{i,1}, \dots, v_{i,m_i}], [X_1, \dots, X_n])$ for $1 \leq i \leq s$.*

Proof: Clearly it is at least as strong. To show strictness, consider $\text{PRECEDENCE}([[1, 2, 3], [4, 5, 6]], [X_1, \dots, X_5])$ with $X_1, X_2, X_3 \in \{1, 2, 3, 4, 5, 6\}$, $X_4 \in \{3\}$ and $X_5 \in \{6\}$. Then $\text{PRECEDENCE}([[1, 2, 3], [4, 5, 6]], [X_1, \dots, X_5])$ is unsatisfiable. However, $\text{PRECEDENCE}([1, 2, 3], [X_1, \dots, X_5])$ and $\text{PRECEDENCE}([4, 5, 6], [X_1, \dots, X_5])$ are both GAC. \diamond

Decomposition is again unnecessary as we can encode the global constraint into a sequence of ternary constraints. The idea is to keep a tuple recording the greatest value used so far within each equivalence class as we slide down the sequence. We introduce $n + 1$ new finite-domain variables, Y_i whose values are s -tuples. We write $Y_i[j]$ to indicate the j th component of the tuple. We then post a sequence of ternary constraints, $E(X_i, Y_i, Y_{i+1})$ for $1 \leq i \leq n$ which hold iff for all $1 \leq j \leq s$ we have $X_i \neq v_{j,k}$ for all $k > Y_i[j] + 1$, $Y_{i+1}[j] = Y_i[j] + 1$ if $X_i = v_{j,Y_i[j]+1}$ and $Y_{i+1}[j] = Y_i[j]$ otherwise. The value taken by $Y_{i+1}[j]$ is the largest index within the j th equivalence class used up to X_i . Since E is functional in its third argument, this takes $O(nds)$ time where $e = \prod_{i \leq s} m_i$. Note that if all values are interchangeable with each other, then $s = 1$ and $m_1 = m$, and enforcing GAC takes $O(nmd)$ time. Similarly, for just one pair of interchangeable values, $s = n - 1$, $m_1 = 2$ and $m_i = 1$ for $i > 1$, and enforcing GAC takes $O(nd)$ time.

6 WREATH VALUE INTERCHANGEABILITY

Wreath value interchangeability [13] is a common type of symmetry in problems where variables are assigned a pair of values from $D_1 \times D_2$, values in D_1 are fully interchangeable, and for a fixed value in D_1 , values in D_2 are interchangeable as well. For example, if we are scheduling a conference, the days of the week might be interchangeable, and given a particular day, the meeting rooms might then be interchangeable. For simplicity, we assume the same precedence ordering is used for the values in D_2 for every fixed value in D_1 . However, we can relax this assumption without difficulty.

We can break all the symmetry of wreath-value interchangeability with the global constraint:

$$\text{PRECEDENCE}([u_1, \dots, u_m, [v_1, \dots, v_p]], [X_1, \dots, X_n])$$

This holds iff $\min\{i \mid X_i[1] = u_j \vee i = n+1\} < \min\{i \mid X_i[1] = u_{j+1} \vee i = n+2\}$ for all $1 \leq j < m$, and $\min\{i \mid X_i = \langle u_j, v_k \rangle \vee i = n+1\} < \min\{i \mid X_i = \langle u_j, v_{k+1} \rangle \vee i = n+2\}$ for all $1 \leq j \leq m$ and $1 \leq k < p$. This can be decomposed into precedence constraints of the form $\text{PRECEDENCE}([\langle u_i, v_j \rangle, \langle u_k, v_l \rangle], [X_1, \dots, X_n])$, but this hinders propagation.

Theorem 3 *Enforcing GAC on $\text{PRECEDENCE}([u_1, \dots, u_m, [v_1, \dots, v_p]], [X_1, \dots, X_n])$ is strictly stronger than enforcing GAC on $\text{PRECEDENCE}([\langle u_i, v_j \rangle, \langle u_k, v_l \rangle], [X_1, \dots, X_n])$ for all $1 \leq i < k \leq m$, and for all $i = j$, $1 \leq i, j \leq m$, $1 \leq k < l \leq p$.*

Proof: Clearly it is at least as strong. To show strictness, consider $\text{PRECEDENCE}([1, 2, [3, 4]], [X_1, X_2, X_3, X_4])$

with $X_1 \in \{(1, 3)\}$, $X_2 \in \{(1, 3), (1, 4)\}$, $X_3 \in \{(1, 3), (2, 3)\}$, $X_4 \in \{(2, 3), (2, 4)\}$. Then enforcing GAC on $\text{PRECEDENCE}([1, 2, [3, 4]], [X_1, X_2, X_3, X_4])$ prunes $(1, 3)$ from X_2 . However, $\text{PRECEDENCE}([\langle u, v \rangle, \langle w, z \rangle], [X_1, X_2, X_3, X_4])$ is GAC for all $1 \leq u < w \leq 2$ and $3 \leq v, z \leq 4$, and for all $u = w$, $1 \leq u, w \leq 2$, $3 \leq v < z \leq 4$. \diamond

We again can propagate such a precedence constraint using a simple encoding into ternary constraints. We have a finite domain variable which records the greatest pair used so far down the sequence. We can then encode $\text{PRECEDENCE}([u_1, \dots, u_m, [v_1, \dots, v_p]], [X_1, \dots, X_n])$ by means of a sequence of ternary constraints, $F(X_i, Y_i, Y_{i+1})$ for $1 \leq i \leq n$ which hold iff $X_i[1] \neq u_j$ for all $j > Y_i[1] + 1$, if $X_i[1] = u_{Y_i[1]}$ then $X_i[2] \neq v_j$ for all $j > Y_i[2] + 1$, $Y_{i+1} = \langle Y_i[1] + 1, 1 \rangle$ if $X_i[1] = u_{Y_i[1]} + 1$ and $Y_i[2] = m$, $Y_{i+1} = \langle Y_i[1], Y_i[2] + 1 \rangle$ if $X_i = \langle u_{Y_i[1]}, v_{Y_i[2]} + 1 \rangle$, and $Y_{i+1} = Y_i$ otherwise. Enforcing GAC using this encoding takes $O(ndmp)$ time. The extension to wreath value partial interchangeability and to wreath value interchangeability over k -tuples where $k > 2$ are both straight forwards.

7 MAPPING INTO VARIABLE SYMMETRY

An alternative way to deal with value symmetry is to convert it into variable symmetry [4, 10]. We introduce a matrix of 0/1 variables where $B_{ij} = 1$ iff $X_i = j$. We assume the columns of this 0/1 matrix represent the interchangeable values, and the rows represent the original finite domain variables. We now prove that value precedence is equivalent to lexicographically ordering the columns of this 0/1 matrix, but that channelling into 0/1 variables hinders propagation.

Theorem 4 $\text{PRECEDENCE}([v_1, \dots, v_m], [X_1, \dots, X_n])$ is equivalent to $X_i = v_j$ iff $B_{i,j} = 1$ for $0 < j \leq m$ and $0 < i \leq n$, and $[B_{1,j}, \dots, B_{n,j}] \geq_{\text{lex}} [B_{1,j+1}, \dots, B_{n,j+1}]$ for $0 < j < m$.

Proof: By induction on n . In the base case, $n = 1$, $X_1 = v_1$, $B_{1,1} = 1$ and $B_{1,j} = 0$ for $1 < j \leq m$. In the step case, suppose the value precedence constraint holds for a ground assignment in which v_k is the largest value used so far. Consider any extension with $X_{n+1} = v_l$. There are two cases. In the first, $l = k + 1$. The l th column is thus $[0, \dots, 0, 1]$. This is lexicographically less than all previous columns. In the other case, $l \leq k$. Adding a new row with a single 1 in the l th column does not change the ordering between the $l - 1$ th, l th and $l + 1$ th columns. The proof reverses in a similar way. \diamond

We could thus impose value precedence by channelling into an 0/1 matrix model and using lexicographical ordering constraints [4]. However, this decomposition hinders propagation as the lexicographical ordering constraints do not exploit the fact that the 0/1 matrix has a single non-zero entry per row. Indeed, even if we add this implied constraint to the decomposition, propagation is hindered. It is thus worth developing a specialized propagator for the global PRECEDENCE constraint.

Theorem 5 GAC on $\text{PRECEDENCE}([v_1, \dots, v_m], [X_1, \dots, X_n])$ is strictly stronger than GAC on $X_i = v_j$ iff $B_{i,j} = 1$ for $0 < j \leq m$ and $0 < i \leq n$, GAC on $[B_{1,j}, \dots, B_{n,j}] \geq_{\text{lex}} [B_{1,j+1}, \dots, B_{n,j+1}]$ for $0 < j < m$, and GAC on $\sum_{j=1}^m B_{i,j} = 1$ for $0 < i \leq n$.

Proof: Clearly it is as strong. To show strictness, consider $X_1 = 1$, $X_2 \in \{1, 2, 3\}$, $X_3 = 3$, $B_{1,1} = B_{3,3} = 1$, $B_{1,2} = B_{1,3} = B_{3,1} = B_{3,2} = 0$, and $B_{2,1}, B_{2,2}, B_{2,3} \in \{0, 1\}$. Then the decomposition is

GAC. However, enforcing GAC on the value precedence constraint will prune 1 and 3 from X_2 . \diamond

It is not hard to show that partial value interchangeability corresponds to partial column symmetry in the corresponding 0/1 matrix model. As with full interchangeability, we obtain more pruning with a specialized propagator than with lexicographical ordering constraints.

8 SURJECTION PROBLEMS

A surjection problem is one in which each value is used at least once. Puget converts value symmetries on surjection problems into variable symmetries by channelling into dual variables which record the first index using a value [11]. For interchangeable values, this gives $O(nm)$ binary symmetry breaking constraints: $X_i = j \rightarrow Z_j \leq i$, $Z_j = i \rightarrow X_i = j$, and $Z_k < Z_{k+1}$ for all $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq k < m$. Any problem can be made into a surjection by introducing m additional new variables to ensure each value is used once. In this case, Puget's symmetry breaking constraints ensure value precedence. However, they may not prune all possible values. Consider $X_1 = 1$, $X_2 \in \{1, 2\}$, $X_3 \in \{1, 3\}$, $X_4 \in \{3, 4\}$, $X_5 = 2$, $X_6 = 3$, $X_7 = 4$, $Z_1 = 1$, $Z_2 \in \{2, 5\}$, $Z_3 \in \{3, 4, 6\}$, and $Z_4 \in \{4, 7\}$. Then all the binary implications are AC. However, enforcing GAC on $\text{PRECEDENCE}([1, 2, 3, 4], [X_1, \dots, X_7])$ will prune 1 from X_2 .

9 SET VARIABLES

We also meet interchangeable values in problems involving set variables. For example, in a set variable model of the social golfers problem in which we assign a set of golfers to the groups in each week, all values are interchangeable. We can break all such symmetry with value precedence constraints. For set variables, $\text{PRECEDENCE}([v_1, \dots, v_m], [S_1, \dots, S_n])$ holds iff $\min\{i \mid (v_j \in S_i \wedge v_k \notin S_i) \vee i = n+1\} < \min\{i \mid (v_k \in S_i \wedge v_j \notin S_i) \vee i = n+2\}$ for all $1 \leq j < k \leq m$ [9]. That is, the first time we distinguish between v_j and v_k (because both values don't occur in a given set variable), we have v_j occurring and not v_k . This breaks all symmetry as we cannot now swap v_j for v_k . Law and Lee again give a specialized propagator for enforcing BC on $\text{PRECEDENCE}([v_j, v_k], [S_1, \dots, S_n])$. We prove here that this decomposition hinders propagation.

Theorem 6 Enforcing BC on $\text{PRECEDENCE}([v_1, \dots, v_m], [S_1, \dots, S_n])$ is strictly stronger than enforcing BC on $\text{PRECEDENCE}([v_i, v_j], [S_1, \dots, S_n])$ for all $1 \leq i < j \leq m$.

Proof: Clearly it is at least as strong. To show strictness, consider $\text{PRECEDENCE}([0, 1, 2], [S_1, S_2, S_3, S_4, S_5])$ with $\{\} \subseteq S_1 \subseteq \{0\}$, $\{\} \subseteq S_2 \subseteq \{1\}$, $\{\} \subseteq S_3 \subseteq \{1\}$, $\{\} \subseteq S_4 \subseteq \{0\}$, and $S_5 = \{2\}$. Then enforcing BC on $\text{PRECEDENCE}([0, 1, 2], [S_1, S_2, S_3, S_4, S_5])$ sets S_1 to $\{0\}$. However, $\text{PRECEDENCE}([i, j], [S_1, S_2, S_3, S_4, S_5])$ is BC for all $0 \leq i < j \leq 2$. \diamond

As with finite domain variables, we do need to introduce a new propagator nor to decompose this global constraint. We view each set variable in terms of its characteristic function (a vector of 0/1 variables). This gives us an n by d matrix of 0/1 variables with column symmetry in the d dimension. Unlike the case with finite domain variables, rows can now have any sum. We can break all such column symmetry with a simple lexicographical ordering constraint [4]. If we use the lex chain propagator [2], we achieve BC on the original value precedence constraint in $O(nd)$ time.

In many constraint solvers, set variables also have restrictions on their cardinality. Unfortunately, adding such cardinality information makes value precedence intractable to propagate.

Theorem 7 Enforcing BC on PRECEDENCE([v_1, \dots, v_m], [S_1, \dots, S_n]) where set variables have cardinality bounds is NP-hard.

Proof: We give a reduction from a 1-in-3 SAT problem in N Boolean variables, x_1 to x_N and M positive clauses. We let $n = 2N + M$, $m = 2N$ and $v_i = i$. To encode the truth assignment which satisfies the 1-in-3 SAT problem, we have $S_{2i} = \{2i, 2i+1\}$ and $\{2i\} \subseteq S_{2i+1} \subseteq \{2i, 2i+1\}$ for $1 \leq i \leq N$. S_{2i+1} will be $\{2i\}$ iff x_i is false and $\{2i, 2i+1\}$ otherwise. The remaining M CSP variables represent the M clauses. Suppose the i th clause is $x_j \vee x_k \vee x_l$. We let $S_{2N+i} \subseteq \{2j+1, 2k+1, 2l+1\}$. Finally, we force S_{2N+i} to take two of the values $2j+1, 2k+1, 2l+1$ from its upper bound. Value precedence only permits this if exactly two out of S_{2j}, S_{2k} and S_{2l} take the set value representing “false”. The global value precedence constraint thus has bound support iff the corresponding 1-in-3 SAT problem is satisfiable. Hence, enforcing BC is NP-hard. \diamond

10 VALUE AND VARIABLE SYMMETRY

In many situations, we have both variable and value symmetry. Can we safely combine together symmetry breaking constraints for variable symmetries and value symmetries? Do we break all symmetry?

INTERCHANGEABLE VARIABLES

Suppose that all n variables and m values are interchangeable. We can safely combine a global value precedence constraint (which breaks all the value symmetry) with a simple ordering constraint (which breaks all the variable symmetry). However, this does not break all symmetry. For example, [1, 2, 2] and [1, 1, 2] are symmetric since inverting the first sequence and permuting 1 with 2 gives the second sequence. However, both sequences satisfy the value precedence and ordering constraints. We can break all symmetry with the global constraint INCREASINGSEQ([X_1, \dots, X_n]) which holds iff $X_1 = v_1, X_{i+1} = X_i$ or ($X_i = v_j$ and $X_{i+1} = v_{j+1}$) for all $0 < i < n$, and $|\{i \mid X_i = v_j\}| \leq |\{i \mid X_i = v_{j+1}\}|$ for all $0 < j < m$. That is, the values and the number of occurrences of each value increase monotonically. We can also have the values increasing but the number of occurrences decreasing. One way to propagate this constraint is to consider the corresponding 0/1 matrix model. The INCREASINGSEQ constraint lexicographically orders the rows and columns, as well as ordering the columns by their sums.

Consider, now, (partially) interchangeable set variables taking (partially) interchangeable values. This corresponds to an 0/1 matrix with (partial) row and (partial) column symmetry. Unfortunately, enforcing breaking all row and column symmetry is NP-hard [1]. We cannot expect therefore to break all symmetry when we have interchangeable set variables and interchangeable values. We can break symmetry partially by lexicographical ordering the rows and columns of the corresponding 0/1 matrix.

MATRIX SYMMETRY

Variables may be arranged in a matrix which has row and/or column symmetry [4]. Lexicographical ordering constraints will break such symmetries. Suppose that values are also (partially) interchangeable.

As lexicographical ordering constraints can be combined in any number of dimensions [4], and as value precedence is equivalent to lexicographically ordering the 0/1 model, we can safely combine value precedence and row and column symmetry breaking constraints.

VARIABLE REFLECTION SYMMETRY

Suppose we have a sequence of $2n$ variables with a reflection symmetry. Then we can break all such symmetry with the lexicographical ordering constraint: $[X_1, \dots, X_n] \leq_{\text{lex}} [X_{2n}, \dots, X_{n+1}]$. For an odd length sequence, we just miss out the middle element. If values are also (partially) interchangeable, we can combine such a reflection symmetry breaking constraint with precedence constraints. Whilst these symmetry breaking constraints are compatible, they do not break all symmetry. For example, [1, 2, 1, 1, 2] and [1, 2, 2, 1, 2] are symmetric since inverting the first sequence and permuting 1 with 2 gives the second sequence. However, both sequences satisfy all symmetry breaking constraints.

VARIABLE ROTATION SYMMETRY

Suppose we have a sequence of n variables with a rotation symmetry. That is, if we rotate the sequence, we obtain a symmetric solution. We can break all such symmetry with the constraints: $[X_1, \dots, X_n] \leq_{\text{lex}} [X_i, \dots, X_n, X_1, \dots, X_{i-1}]$ for $1 < i \leq n$. If values are also (partially) interchangeable, then we can combine such symmetry breaking constraints with precedence constraints. Whilst these symmetry breaking constraints are compatible, they do not break all symmetry. For example, [1, 1, 2, 1, 2] and [1, 2, 1, 2, 2] are symmetric since rotating the first sequence by 2 elements and permuting 1 with 2 gives the second sequence. However, both sequences satisfy all symmetry breaking constraints.

11 EXPERIMENTAL RESULTS

To test the efficiency and effectiveness of these encodings of value precedence constraints, we ran a range of experiments. We report results here on Schur numbers (prob015 in CSPLib). This problem was used by Law and Lee in the first experimental study of value precedence [9]. We have observed similar results in other domains like the social golfers problem and Ramsey numbers (prob017 in CSPLib).

The Schur number $S(k)$ is the largest integer n for which the interval $[1, n]$ can be partitioned into k sum-free sets. S is sum-free iff $\forall a, b, c \in S : a \neq b + c$. Schur numbers are related to Ramsey numbers, $R(n)$ through the identity: $S(n) \leq R(n) - 2$. Schur numbers were proposed by the famous German mathematician Isaai Schur in 1918. $S(4)$ was open until 1961 when it was first calculated by computer. $S(3)$ is 13, $S(4)$ is 44, and $160 \leq S(5) \leq 315$. We consider the corresponding decision problem, $S(n, k)$ which asks if the interval $[1, n]$ can be partitioned into k sum-free sets. A simple model of this uses n finite domain variables with k interchangeable values.

Results are given in Table 1. The model *all* uses a single global precedence constraint to break all value symmetry. The model *adjacent* uses the method proposed by Law and Lee in [9] which posts $O(k)$ precedence constraints between adjacent interchangeable values. The model *none* use no precedence constraints. We coded the problem using the finite domain library in SICSTUS 3.12.3, and ran it on an AMD Athlon Dual Core 2.2GHz processor with 1 GB RAM.

The results show the benefits of a global value precedence constraint. With a few interchangeable values, we see the same pruning using *adjacent* as *all*. However, we observe better runtimes with the

problem $S(n, k)$	value symmetry breaking								
	none			adjacent values			all values		
	c	b	t	c	b	t	c	b	t
$S(13, 3)$	126	276	0.02	360	46	0.01	243	46	0.01
$S(13, 4)$	126	134400	16.43	477	2,112	1.48	243	2,112	0.85
$S(13, 5)$				777	210,682	20.80	243	6,606	11.88
$S(13, 6)$				879	309,917	79.60	243	1,032	42.51
$S(14, 3)$	147	456	0.02	399	76	0.02	273	76	0.02
$S(14, 4)$	147	46,1376	39.66	525	8,299	3.06	273	8,299	1.96
$S(14, 5)$				816	813,552	66.83	273	58,558	40.35
$S(14, 6)$				1,731	250,563	348.06	273	57,108	197.39
$S(15, 3)$	168	600	0.03	438	100	0.02	303	100	0.02
$S(15, 4)$	168	1,044,984	101.36	573	17,913	7.92	303	17,913	4.73
$S(15, 5)$				855	1,047,710	259.15	303	194,209	145.97
$S(15, 6)$									

Table 1. Decision problem associated with Schur numbers: constraints posted, backtracks and times to find all solutions in seconds to $S(n, k)$. Blank entries are for problems not solved within the 10 minute cut off. Results are similar to find first solution.

all model as it encodes into fewer ternary constraints ($O(n)$ versus $O(nk)$). With more interchangeable values (e.g. $k > 4$), we observe both better runtimes and more pruning with the single global precedence constraint in the *all* model. The encoding of this global constraint into ternary constraints appears therefore to be an efficient and an effective mechanism to deal with interchangeable values.

12 RELATED WORK

Whilst there has been much work on symmetry breaking constraints for variable symmetries, there has been less on value symmetries. Law and Lee formally defined value precedence [9]. They also gave specialized propagators for breaking value precedence for a pair of interchangeable values. Gent proposed the first encoding of value precedence constraint [6]. However, it is uncertain what consistency is achieved as the encoding indexes with finite domain variables.

A number of methods that modify the underlying solver have been proposed to deal with value symmetry. Van Hentenryck *et al.* gave a labelling schema for breaking all symmetry with interchangeable values [7]. Inspired by this method, Roney-Dougal *et al.* gave a polynomial method to construct a GE-tree, a search tree without value symmetry [12]. Finally, Sellmann and van Hentenryck gave a $O(nd^{3.5} + n^2d^2)$ dominance detection algorithm for breaking all symmetry when both variables and values are interchangeable [13].

There are a number of earlier (and related) results about the tractability of symmetry breaking. Crawford *et al.* prove that breaking all symmetry in propositional problems is NP-hard in general [3]. Bessiere *et al.* prove that the special case of breaking all row and column symmetry for variables in a matrix model is NP-hard [1]. Sellmann and van Hentenryck prove a closely related result that dominance detection for breaking all symmetry with set variables and values that are interchangeable is NP-hard [13].

13 CONCLUSIONS

We have presented a detailed study of the use of value precedence constraints to break value symmetry. We first gave a simple encoding of value precedence into ternary constraints that is both efficient and effective. We then extended value precedence to deal with a number of generalizations like wreath value and partial interchangeability.

We have also shown how value precedence is closely related to lexicographical ordering. Finally, we considered the interaction between value precedence and other symmetry breaking constraints. There are a number of interesting open questions. For example, how does value precedence interact with variable and value ordering heuristics?

REFERENCES

- [1] C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh, ‘The complexity of global constraints’, in *Proc. of the 19th National Conf. on AI*. AAAI, (2004).
- [2] M. Carlsson and N. Beldiceanu, ‘Arc-consistency for a chain of lexicographic ordering constraints’, Technical report T2002-18, Swedish Institute of Computer Science, (2002).
- [3] J. Crawford, G. Luks, M. Ginsberg, and A. Roy, ‘Symmetry breaking predicates for search problems’, in *Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning*, (KR ’96), pp. 148–159, (1996).
- [4] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh, ‘Breaking row and column symmetry in matrix models’, in *8th Int. Conf. on Principles and Practices of Constraint Programming* (CP-2002). Springer, (2002).
- [5] A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh, ‘Global constraints for lexicographic orderings’, in *8th Int. Conf. on Principles and Practices of Constraint Programming* (CP-2002). Springer, (2002).
- [6] I.P. Gent, ‘A symmetry breaking constraint for indistinguishable values’, in *Proc. of 1st Int. Workshop on Symmetry in Constraint Satisfaction Problems (SymCon-01), held alongside CP-01*, (2001).
- [7] P. Van Hentenryck, M. Agren, P. Flener, and J. Pearson, ‘Tractable symmetry breaking for CSPs with interchangeable values’, in *Proc. of the 18th IJCAI*. (2003).
- [8] Y.C. Law, *Using Constraints to Break Value Symmetries in Constraint Satisfaction Problems*, Ph.D. dissertation, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2005.
- [9] Y.C. Law and J.H.M. Lee, ‘Global constraints for integer and set value precedence’, in *Proc. of 10th Int. Conf. on Principles and Practice of Constraint Programming* (CP2004), pp. 362–376. Springer, (2004).
- [10] Y.C. Law and J.H.M. Lee, ‘Breaking value symmetries in matrix models using channeling constraints’, in *Proc. of the 20th Annual ACM Symposium on Applied Computing* (SAC-2005), pp. 375–380, (2005).
- [11] J-F. Puget, ‘Breaking all value symmetries in surjection problems’, in *Proc. of 11th Int. Conf. on Principles and Practice of Constraint Programming* (CP2005), ed., P. van Beek. Springer, (2005).
- [12] C. Roney-Dougal, I. Gent, T. Kelsey, and S. Linton, ‘Tractable symmetry breaking using restricted search trees’, in *Proc. of ECAI-2004*. IOS Press, (2004).
- [13] M. Sellmann and P. Van Hentenryck, ‘Structural symmetry breaking’, in *Proc. of 19th IJCAI*. (2005).

3. Distributed AI/Agents

This page intentionally left blank

Coordination through Inductive Meaning Negotiation

Alessandro Agostini¹

Abstract. This paper is on negotiation, precisely on the negotiation of *meaning*. We advance and discuss a formal paradigm of coordination and variants thereof, wherein meaning negotiation plays a major role in the process of convergence to a common agreement. Our model engages a kind of pairwise, model-theoretic coordination between knowledge-based agents, eventually able to communicate the *complete & local* meaning of their beliefs by expressions taken from the literals of a common first-order language. We focus on the framework of *inductive inference*—sometimes called “formal learning theory,” and argue that it offers a fresh and rigorous perspective on many current debates in Artificial Intelligence in the context of multiple agents in interaction.

1 INTRODUCTION

Following [6], automated negotiation research can be considered to deal with three broad topics, namely: (a) *Negotiation Protocols*: the set of “rules” that govern the interaction among the agents in play. (b) *Negotiation Objects*: the range of issues, or attributes, over which agreement must be reached. (c) *Agents’ Decision Making Models*: the decision making functionality the agents concerned employ to act in line with the negotiation protocol (a) in order to achieve their objectives (b). References to the literature on negotiation protocols and objects are, for instance, [15, 6, 19, 17] and the references cited there. Either negotiation protocols together with negotiation objects or agents’ decision making models is the dominant concern; it depends on the negotiation scenario. In this paper we are interested in a unique negotiation object: *meaning*.

Meaning negotiation is an inductive process that is shaped by multiple elements and that affects these elements. In the process, negotiating meaning entails both interpretation and action. In fact, what we are going to present is a perspective of meaning negotiation that does not imply a fundamental distinction between interpreting and acting, doing and thinking, or understanding and responding. All are part of the ongoing process of the same ‘game’, and the game always generates new circumstances for further negotiation and further meanings. To illustrate, consider the following coordination scenario [7].

Example 1 “Suppose that Frank and Gertrude have to pass one another in the hall every day. On the first day, Frank thinks Gertrude will move to her right, so he does the same. But Gertrude moves to her left. Frank assumes that her convention is to always move to the left, so he moves to his left. But Gertrude draws the symmetrical inference and moves to her right. Thinking that this has gone far enough, Frank moves to his right again, resolving never to move so that Gertrude can get past. But Gertrude, equally exasperated, makes the same decision! The two die of starvation in the hallway.” (p. 267)

¹ Department of Information and Communication Technology, University of Trento, I-38100, Italy. Email: agostini@dit.unitn.it.

Think of the mutual history of Frank and Gertrude as a sequence of successive attempts or trials at passing *at the semantic level*. This means, intuitively, that a sequence of meanings has been generated from previous actions, and that a sequence of actions has been generated from previous meanings. Since they will continue to have “rites of passage” for the unforeseeable future, it would certainly be nice if they could eventually reach a “state of knowledge” in which they always pass on the first attempt. We refer such state of knowledge as the agents’ “common meaning.” How would we model such stable state of knowledge? Can we characterize games like this for which a common meaning exists?

In this paper, we present a formal framework suitable to investigate questions like these. Characterization results, in particular, are as general as important, interesting and, sometimes, difficult to deserve future investigation within the framework we advance in this paper. A main feature of our framework is that the relation between autonomous agents’ actions and preferences leading to actions is made explicit. For doing this, we use inductive logic, as we will see. More precisely, we propose a new approach to the study of coordination between knowledge-based agents, eventually able to explicitly employ the *local* meaning of their beliefs or actions² to solve their coordination problem. In our approach, meaning negotiation plays an important role, as it seems to be a fruitful way to overcome the problem of the local-to-the-agent association between linguistic forms—that represent either physical actions or “speech acts”—and their meaning. We proceed along the model-theoretic tradition of *inductive inference*—say [7, 10] and the references cited there, that descends from the pioneering studies on inductive inference developed by Solomonoff, Putman, Gold, Blum, among others (see for instance [5] for a survey).

This paper is thus structured as follows. Below is an intuitive game-theoretic picture of our basic model of coordination through meaning negotiation, followed (Section 3) by the formal presentation of our model. In Section 4 we illustrate the behavior of a special kind of “self-centered agents,” and give a result on their coordination competence. We conclude with some related work (Section 5), and a summary together with some future work (Sections 6,7).

2 A GAME FOR MEANING NEGOTIATION

We imagine a game of two “knowledge-based” agents, say Alfonso and Barbara, whose knowledge is represented by a nonempty class of structures, **A** and **B**, that interpret (locally) a common (global) lexicon \mathcal{L} . Alfonso and Barbara have preferences and beliefs in the following sense: Barbara believes a sentence θ of \mathcal{L} if there is a structure in **B** where θ holds. Given two sentences θ, ϑ of \mathcal{L} , Barbara prefers

² In this paper we consider beliefs as actions and vice versa. Otherwise, coordination cannot occur, cf. Def. 3ab. This seems true in the passing game.

θ to ϑ if for every structure \mathcal{A} in \mathbf{B} which satisfies θ and every structure \mathcal{B} in \mathbf{B} which satisfies ϑ , the cardinality of \mathcal{A} 's domain is not higher than the cardinality of \mathcal{B} 's domain. Intuitively, under this interpretation of preferences, we say that agents prefer statements which require to employ the most economic structures to convey meaning.

To coordinate, Alfonso and Barbara communicate with each other in order to respectively end up in the limit with a consistent description of a structure, $\mathcal{A} \in \mathbf{A}$ and $\mathcal{B} \in \mathbf{B}$, such that \mathcal{A} is sufficiently close to \mathbf{B} and \mathcal{B} is sufficiently close to \mathbf{A} . We might think that \mathcal{A} is sufficiently close to \mathbf{B} if every finite set of statements expressed by Alfonso (so true in \mathcal{A} , recall footnote 2 on actions as beliefs) is satisfiable in some $\mathcal{B}' \in \mathbf{B}$, and vice versa. Thus, we expect that the more Alfonso is like Barbara, that is, Alfonso and Barbara have similar preferences, the better chance Alfonso and Barbara have to coordinate. The sets of sentences K_A and K_B eventually satisfied by \mathcal{A} and \mathcal{B} , respectively, are Alfonso and Barbara's knowledge base at the specific, temporal point t of their interaction. If satisfied by \mathcal{A} , we say that K_A is the *theory* played by Alfonso up to t . To dramatize, let us suppose that each agent does not know the preferences of the other agent—that is, what structure it will eventually play.

To start the game, Alfonso is conceived as choosing one member from \mathbf{A} to be his “actual world”, namely, the structure that models Alfonso's initial preferences. Alfonso's choice is initially unknown to Barbara. Alfonso then provides a “clue” (Alfonso's action) about his chosen world. At the same time, Barbara does her choice as well, and provides Alfonso with a clue about her actual world. We assume that Alfonso and Barbara are allowed to change their actual world (“mind change” in the terminology of inductive inference [5]) at each step of the game, provided that they remain coherent to the actions shown upon some point and change their mind finitely often. Clearly, it is safe to begin with a sequence of actions (a *behavior*) coherent to the majority of the structures in each class. In this way, if Alfonso realizes that the structure he has in mind, *i.e.* his actual world, is not close enough to Barbara's actual world, he can change it, and this holds for Barbara as well.

Alfonso and Barbara are allowed to start the game again from the beginning whenever some disagreement occurs. Of course, to reach coordination disagreement should happen only finitely often.

Alfonso and Barbara succeed in their coordination game if each player's actual world becomes in the limit sufficiently close to the preferred beliefs played by the other agent, in the sense explained.

Remark 1 It is important to note that, although the game just depicted refers to a first-order setting,—we present it formally in the next section—, our approach to the modeling of coordination may be well generalized to a number of other semantic paradigms, where language \mathcal{L} and structures are not first-order. The central point is that we have to capture (formally) the semantics of the language in use by the agents, and that such language is build upon a common vocabulary. (Otherwise, agents cannot even “understand” each other at the *syntactic* level!) We have used structures in the mathematical sense. Example 3 gives an application scenario where structures are to be thought mostly of “schemas” or ontologies.

Remark 2 By assuming the agents be computable machines, the pair (Alfonso, Barbara) is a kind of *interactive proof system* as defined, for instance, in complexity theory [13]. Moreover, note that the form of interaction between the two agents may differ according to what situation one aims to model. We have chosen a purely alternating interaction with first move by Alfonso and synchronous moves (cf. subsection 3.2.3). Other choices of dynamics are possible with slight technical modifications.

3 THE PARADIGM

In this section we formalize and discuss the foregoing ideas. As we deal with structures, or models, from now on we refer to the resulting formal paradigm as *model-coordination* (*m*-coordination in short).

3.1 Preliminaries and Notation

Sequences. We denote the set $\{0, 1, 2, \dots\}$ of natural numbers by N . Let η be an infinite sequence. For $i \in N$, we write $\eta|_i$ for the proper initial sequence of length i in η . For every finite or infinite sequence ζ of length $n > k$, $k \in N$, we let $\zeta[k]$ denote the finite sequence $\langle \zeta_0 \cdots \zeta_k \rangle$ and $\zeta|k$ denote the sequence obtained from ζ by deleting its first $k + 1$ elements. We write $|\sigma|$ for the length of a finite sequence, and \emptyset for the finite sequence of length zero. Thus, $\zeta[k] = \zeta|_{k+1}$, and $\emptyset|\zeta = \emptyset$ if $|\zeta| = 1$. We write σ_i or also $(\sigma)_i$ for the i th element of σ , $0 \leq i < |\sigma|$. We denote the set of elements in a (finite, infinite) sequence τ by *range*(τ).

A First-Order Framework. We consider a first-order language with equality \mathcal{L} and write \mathcal{L}_{basic} to denote the set of literals of \mathcal{L} . The members of \mathcal{L}_{basic} are called *basic formulas*. Our semantic notions are standard. Let \mathcal{S} be a structure that interprets \mathcal{L} . Let \models denote the model theoretic concept of truth in a structure. Then \mathcal{S} is a *model* of $\Gamma \subseteq \mathcal{L}_{basic}$, and Γ is said to be *satisfiable* in \mathcal{S} , if there is an assignment $h : \text{Var} \rightarrow \text{dom}(\mathcal{S})$ with $\mathcal{S} \models \Gamma[h]$. Γ is *satisfiable* if it is satisfiable in some structure. An assignment h to \mathcal{S} is *complete* if h is a mapping onto $\text{dom}(\mathcal{S})$. The class of models of Γ is denoted by $\text{MOD}(\Gamma)$.

3.2 Basic components

The basic components of model-coordination and the criterion of coordination success are now to be introduced formally.

3.2.1 Environments

Let SEQ denote the collection of all finite sequences over the set \mathcal{L}_{basic} of atomic formulas on vocabulary \mathcal{L} and their negations. We define an *environment* to be any infinite sequence over \mathcal{L}_{basic} . Thus, for all $\sigma \in SEQ$, there is an environment e such that $\sigma = e|_n$ with $n = |\sigma|$. In this technical sense, SEQ then denotes the collection of all proper initial sequences of any environment. To consider consistent data-streams, we need to relate them to a structure. We do this in the next definition.

Definition 1 Let structure \mathcal{S} and complete assignment h to \mathcal{S} be given. An environment e is *for* \mathcal{S} via h just in case $\text{range}(e) = \{\beta \in \mathcal{L}_{basic} \mid \mathcal{S} \models \beta[h]\}$. An environment e is *for* \mathcal{S} just in case e is an environment for \mathcal{S} via some complete assignment.

In other words, an environment for a structure \mathcal{S} via complete assignment h lists the basic diagram of \mathcal{S} under h . (The *basic diagram* of \mathcal{S} under complete assignment h is the subset of \mathcal{L}_{basic} made true in \mathcal{S} via h .)

3.2.2 Agents

An agent in the model-coordination paradigm, or *basic agent*, is a pair (Ψ, \mathbf{A}) , where Ψ is any (computable, total) mapping of SEQ into \mathcal{L}_{basic} , and \mathbf{A} is a nonempty class of countable structures. Faced with data $\sigma \in SEQ$, a basic agent (Ψ, \mathbf{A}) *believes the action* $\Psi(\sigma)$ if there

is $S \in \mathbf{A}$ such that $S \models \Psi(\sigma)[h]$ for some (complete) assignment h to S . If this is the case, we call $\Psi(\sigma)$ a *belief* and the pair S, h a *background knowledge* of $\langle \Psi, \mathbf{A} \rangle$, denoted: $\{\mathcal{S}, h\}_{ag}$ with $ag = \langle \Psi, \mathbf{A} \rangle$. The first component Ψ is called *communication function*, or “ability,” by it the agent generates its actions. The second component \mathbf{A} is called *background world*. We say that $\langle \Psi, \mathbf{A} \rangle$ is *based on* \mathbf{A} . We write Λ^b for the class of all basic agents and $\Lambda^b(\mathbf{A})$ for all basic agents based on \mathbf{A} .

3.2.3 Dynamics

We now consider formally the kind of interaction, or “protocol,” between two basic agents. In this paper, we restrict attention to dynamics based on simultaneous moves, that is, the agents make decisions at the same time, and to pairwise communication only, that is, interactions involve just two agents. Our presentation may be generalized to a number of different protocols—examples are sequential moves, n -person communication, explicit guesses, etcetera. It only depends on what the specific scenario or application requires, cf. Remark 2.

Definition 2 Let basic agents $\langle \Psi, \mathbf{A} \rangle$ and $\langle \Phi, \mathbf{B} \rangle$ be given.

1. The *interaction sequence* (or “play”) of $\langle \Psi, \mathbf{A} \rangle$ and $\langle \Phi, \mathbf{B} \rangle$ is the infinite sequence

$$D_{\Psi, \Phi} = (\langle \overline{\Psi}_i, \overline{\Phi}_i \rangle : i \in N),$$

where $\overline{\Psi}_i$ is the i th move of Ψ and $\overline{\Phi}_i$ is the i th move of Φ , defined by induction as follows.

- i. $\overline{\Psi}_0 = \Psi(\emptyset)$ and $\overline{\Phi}_0 = \Phi(\emptyset)$.
- ii. $\overline{\Psi}_{n+1} = \Psi(\overline{\Phi}[n])$ and $\overline{\Phi}_{n+1} = \Phi(\overline{\Psi}[n])$.
2. The *response sequence*

$$\overline{\Psi} = (\overline{\Psi}_i : i \in N)$$

is the sequence of moves by basic agent Ψ in response to basic agent Φ , and the *response sequence*

$$\overline{\Phi} = (\overline{\Phi}_i : i \in N)$$

is the sequence of moves by Φ in response to Ψ .

3. Let $k \in N$ be given. The *interaction sequence* of $\langle \Psi, \mathbf{A} \rangle$ and $\langle \Phi, \mathbf{B} \rangle$ starting at k is the infinite sequence

$${}^k D_{\Psi, \Phi} = (\langle {}^k \overline{\Psi}_i, {}^k \overline{\Phi}_i \rangle : i \in N),$$

where ${}^k \overline{\Psi}_i$ is the i th element in ${}^k \overline{\Psi}$ and ${}^k \overline{\Phi}_i$ is the i th element in ${}^k \overline{\Phi}$.

We then say that ${}^k \overline{\Psi}_i$ is the i th move of Ψ starting at k and ${}^k \overline{\Phi}_i$ is the i th move of Φ starting at k . Notice that $\overline{\Psi}$ is finite if and only if at any interaction step $i \in N$, $\Phi(\overline{\Psi}|_i)$ or $\Psi(\overline{\Phi}|_i)$ is undefined. If this is the case, it is immediate to verify that also $\overline{\Phi}$ is finite. Observe that $D_{\Psi, \Phi} = \langle \overline{\Psi}_0, \overline{\Phi}_0 \rangle^0 D_{\Psi, \Phi}$.

Remark 3 In our paradigm, the agents make decisions independently at any stage of the coordination process, no agent being fully informed of the choice of the other agent prior to making his own decision. The choice of meaning is always local and hidden to the other agent. Our model of cooperation is a kind of “extensive game with imperfect information and sequential equilibria” [12].

3.2.4 Success

To coordinate agents have to negotiate in the limit a ‘complete and consistent’ description of a structure in their own background world, eventually after a finite number of disagreements. The agents can restart their interaction finitely often, but after the last disagreement they must eventually coordinate. We rely on the following definition.

Definition 3 Let basic agents $\langle \Psi, \mathbf{A} \rangle$ and $\langle \Phi, \mathbf{B} \rangle$ be given. We say that $\langle \Psi, \mathbf{A} \rangle$ *m-coordinates with* $\langle \Phi, \mathbf{B} \rangle$ (written: $\Psi \rightleftharpoons_m \Phi$) just in case for some $s, t \in N$:

- a $_s|\overline{\Psi}$ is an environment for some $\mathcal{A} \in \mathbf{A}$,
- b $_t|\overline{\Phi}$ is an environment for some $\mathcal{B} \in \mathbf{B}$, and
- c for all $n \in N$, $_s|\overline{\Psi}|_n$ is satisfiable in some $\mathcal{B}' \in \mathbf{B}$ and $_t|\overline{\Phi}|_n$ is satisfiable in some $\mathcal{A}' \in \mathbf{A}$.

Clauses a, b fix the relation between actions and beliefs that we call *coherence* (of agents’s behavior). So, there is no coordination without coherence. Note that the definition depends on the interaction sequence $D_{\Psi, \Phi}$, and that structures \mathcal{A}' and \mathcal{B}' depend on n . For simplicity, however, the reader can relax the definition by assuming $\mathcal{A}' = \mathcal{A}$ and $\mathcal{B}' = \mathcal{B}$. In such case, this means that agents coordinate if and only if each agent is able to choose and *enumerate* (cf. Def. 3a,b) a structure among those available, which is a substructure of the structure choosen by the other agent. So, to *m*-coordinate each basic agent outputs in the limit the basic diagram (under some complete assignment) of a structure in his own background world, and such basic diagram must be finitely consistent in some structure of the other agent’s background world—generally different from the structure described by the first agent, but possibly the same.

Example 2 Let us go back to the passing game of Example 1 and compare it with the definition of “success” by *m*-coordination. In the passing game, a stable state of knowledge (equilibrium) is a pair of plays (e.g., (left, left)) such that given Frank has played the first, Gertrude would not be inclined to deviate from the second, and given that Gertrude has played the second, Frank would not be inclined to deviate from the first. Hence, (left, left) and (right, right) are both the equilibria. In our much richer model, we may think of each basic agent with one structure S with domain of two elements, one element interpreting “go left”, the other element interpreting “go right.” Both agents are allowed to use autonomously their background knowledge, respectively $\{\mathcal{S}, h\}_{Frank}$ and $\{\mathcal{S}, g\}_{Gertrude}$ for h, g complete assignments to S . Clearly, the passing game is a special case of model-coordination with $\mathbf{A} = \mathbf{B} = \{\mathcal{S}\}$, where the existence of natural numbers s, t of Definition 3 is not guaranteed, since the players simply do not find out a common assignment $a = h = g$ of their actions. (Frank and Gertrude “die of starvation in the hallway.”)

The next theorem shows that *m*-coordination is indeed possible, and that a game where an “equilibrium” in the sense of the passing game exists. We rely on the following definition of game.

Definition 4 Let nonempty sets $\Sigma(\mathbf{A})$ and $\Sigma(\mathbf{B})$ of basic agents based on, respectively, \mathbf{A} and \mathbf{B} be given. Let $\{D_{\Psi, \Phi}\} = \{D_{\Psi, \Phi} | \langle \Psi, \mathbf{A} \rangle \in \Sigma(\mathbf{A}), \langle \Phi, \mathbf{B} \rangle \in \Sigma(\mathbf{B})\}$. A *meaning negotiation (m-negotiation) game* is a triple $(\Sigma(\mathbf{A}), \Sigma(\mathbf{B}), \{D_{\Psi, \Phi}\})$. An *equilibrium* of a *m*-negotiation game is a pair $(\{\mathcal{A}, h\}_a, \{\mathcal{B}, g\}_b)$ of background knowledges that satisfies conditions a–c of Definition 3 for some $a \in \Sigma(\mathbf{A})$, $b \in \Sigma(\mathbf{B})$. A *partial equilibrium* (of a *m*-negotiation game) is a pair $(\{\mathcal{A}', h\}_a, \{\mathcal{B}', g\}_b)$ of background knowledges that satisfies condition c of the definition.

The basic idea behind the notion of “equilibrium” is that an equilibrium should specify not only the basic agents’ ability in the limit but also their final beliefs about the coordination history that occurred. The notion of partial equilibrium is less stringent; it specifies the agents’ beliefs at *each* stage of the coordination process. Of course, an equilibrium and a partial equilibrium coincide as a special case. Note also that neither an equilibrium nor a partial equilibrium of a m -negotiation game is necessarily unique. But while an equilibrium does not depend on game stages, a partial equilibrium does.

Theorem 1 Suppose that \mathcal{L} is limited to a binary predicate symbol R . Let \mathbf{A} be the class of all countable linear orders with maximum that interpret R , and let \mathbf{B} be the class of all countable linear orders without maximum that interpret R . Then, there are abilities Ψ, Φ such that $\langle \Psi, \mathbf{A} \rangle$ m -coordinates with $\langle \Phi, \mathbf{B} \rangle$.

Proof: Observe that any initial segment of any environment for a linearly ordered structure (with or without maximum) is satisfiable in any infinite linearly ordered structure, since it generates a substructure of the infinite linearly ordered structure and basic formulas are preserved in overstructures [4, Thm. 2.4.1]. Define Ψ and Φ such that Ψ is an environment for some infinite structure in \mathbf{A} and Φ is an environment for some infinite structure in \mathbf{B} . ■

A less abstract example follows.

Example 3 (Schema-Matching Problem) We define a schema a pair of the form (\mathcal{S}, π) , where \mathcal{S} is a structure and π is a set of sentences satisfiable in \mathcal{S} . We can represent the following problem of “schema matching” [9, 3]. Given two schemas (or “classifications”) (\mathcal{A}, π_1) and (\mathcal{B}, π_2) with their associated data, for each $\vartheta \in \pi_1$ (i.e., “concept”), find the most similar $\vartheta' \in \pi_2$, for a predefined similarity measure. Then, m -coordination provides a similarity measure in the strict sense of the existence of two basic agents $\langle \Psi, \mathcal{A} \cup \text{MOD}(\pi_1) \rangle$ and $\langle \Phi, \mathcal{B} \cup \text{MOD}(\pi_2) \rangle$ who m -coordinate. Less strict similarity measures are possible as variants of m -coordination.³

However, not every m -negotiation game has an equilibrium, as shows the following example of a game where players have diametrically opposed interests—it is called ‘strictly competitive’ by game theorists.

Example 4 Suppose that \mathcal{L} is limited to unary predicate symbols H and T [read H “Head” and T “Tail”]. Let $\theta \in \mathcal{L}_{\text{sen}}$ be of the form: $\forall x(Tx \leftrightarrow \neg Hx)$ [read just one between “Head” and “Tail” occurs], $\vartheta \in \mathcal{L}_{\text{sen}}$ be of the form: $\forall x Tx$ [read “Tail” occurs], and $\delta \in \mathcal{L}_{\text{sen}}$ be of the form: $\forall x Hx$ [read “Head” occurs]. Let $\mathbf{A} = \text{MOD}(\{\theta, \vartheta\})$ and let $\mathbf{B} = \text{MOD}(\{\theta, \delta\})$. It is clear that the m -negotiation game $\langle \Lambda^b(\mathbf{A}), \Lambda^b(\mathbf{B}), \{D_{\Psi, \Phi}\} \rangle$ has no equilibrium, since no pair $\langle \Psi, \mathbf{A} \rangle, \langle \Phi, \mathbf{B} \rangle$ of basic agents can m -coordinate.

4 COMPETENCE OF BASIC AGENTS

A general question to ask about coordination concerns the existence of classes of agents such that the coordination competence of each agent in the class cannot be strictly improved. As a kind of evaluation of the representation power of our framework, although somewhat technical, in this section we exhibit a class of “self-centered” basic agents that has the required property for m -coordination.

³ Because of the importance of schema matching in a number of AI-related research areas, we deserve a full study of it as an independent problem to future work. Here we only showed how the problem can be formalized within our paradigm of m -coordination.

Definition 5 We say that basic agent $\langle \Psi, \mathbf{A} \rangle$ is *self-centered* just in case for every $\Phi : \text{SEQ} \rightarrow \mathcal{L}_{\text{basic}}$, there is $\mathcal{A} \in \mathbf{A}$ such that $R(\Psi, \Phi)$ is an environment for \mathcal{A} .

In other words, a self-centered basic agent enumerates with some other basic agent an environment for some structure in the background world the agent is based on.

We now show (Corollary 1) that the negotiation competence of any total, self-centered basic agent cannot be improved by any total basic agent with the same background knowledge. A *total* basic agent is a basic agent with a total communication function. We rely on the following sense of “meaning negotiation competence.”

Definition 6 The *m -negotiation competence* of a basic agent $\langle \Psi, \mathbf{A} \rangle$ is the set

$$m\text{-scope}(\Psi_{\mathbf{A}}) = \{\langle \Phi, \mathbf{B} \rangle \in \Lambda^b \mid \Psi \Rightarrow_m \Phi\}.$$

Definition 7 Basic agents $\langle \Psi, \mathbf{A} \rangle, \langle \Phi, \mathbf{A} \rangle$ are *distinct* just in case $\Psi(\sigma) \neq \Phi(\sigma)$ for some $\sigma \in \text{SEQ}$.

Definition 8 Let $\sigma \in \text{SEQ}$ and basic agent $\langle \Psi, \mathbf{A} \rangle$ be given. We say that $\langle \Psi, \mathbf{A} \rangle$ (or also Ψ) *starts with* σ just in case for all $\tau \in \text{SEQ}$ with $|\tau| < |\sigma|$, $\Psi(\tau) = (\sigma)|_{|\tau|}$.

Theorem 2 For all distinct, self-centered basic agents $\langle \Psi, \mathbf{A} \rangle$ and $\langle \Psi', \mathbf{A} \rangle$, there is a basic agent such that $\langle \Psi, \mathbf{A} \rangle$ m -coordinates with and that $\langle \Psi', \mathbf{A} \rangle$ does not.

The proof is by construction of a basic agent $\langle \Phi, \mathbf{A} \rangle$. We proceed informally. (See [1, Prop. 2.(49)] for a formal proof.) The meaning negotiation between $\langle \Phi, \mathbf{A} \rangle$ and the agents in Theorem 2 runs as follows. At step 0, Φ starts with some $\sigma \in \text{SEQ}$. Then, if there is no previous move by any opponent basic agent to look at, Φ moves “safe”. Otherwise, if an opponent basic agent “agrees” with basic agent $\langle \Psi, \mathbf{A} \rangle$ on σ , then Φ starts copying the opponent’s moves from the first one. If the opponent basic agent “disagrees” with $\langle \Psi, \mathbf{A} \rangle$ on σ , then Φ breaks off coordination and starts producing inconsistent information.

Corollary 1 Let $\langle \Psi, \mathbf{A} \rangle$ be any total self-centered basic agent. Then there is no total basic agent $\langle \Psi', \mathbf{A} \rangle$ such that $m\text{-scope}(\Psi_{\mathbf{A}}) \subset m\text{-scope}(\Psi'_{\mathbf{A}})$.

That is, the meaning negotiation competence of a total, self-centered basic agent cannot be strictly improved.

5 DISCUSSION AND RELATED WORK

Even though our paradigm omits many important aspects of human coordination and computational systems, it helps illuminate a wide range of phenomena. Our paradigm may help analyze design alternative for distributed systems of “peers” [14] and they suggest ways of analyzing the structural changes associated with introducing the inductive approach to coordination into multiagent systems. On the other hand, a limit of our paradigm is the requirement that “to coordinate agents have to negotiate a ‘complete and consistent’ description of a structure.” Can’t there be partial matches, partially consistent descriptions, or even, in fact, totally inconsistent descriptions that lead to coordination? The answer is ‘yes’. Some initial results and comparison thereof can be found in [1, Secs 2.4-2.5]. For an extension of such results to meaning negotiation, see Section 7 on future work.

Even if restricting the study of coordination to a first-order setting—as we did, the variety of coordination paradigms is huge. Related topics, among others, are discussed in the research on coordination, argumentation and dialogue games. We refer to [2, 11] and the references cited there. In particular, our model-coordination seems to capture in a formal way many ideas discussed informally in [11], for instance the notion of “environment,” and the interactions the agents have therein.

Our approach is close in spirit, if not in formal development, to the framework of the *language games* [18], namely, models of language change and language evolution in populations of communicating agents. Some references are be found in [16]. Language games, however, focus much more on language creation and evolution than our models. Moreover, in Remark 1 we have noted that the agents’ language is build upon a common vocabulary. This is not necessarily the case. In [8], for instance, a mapping of differing syntaxes among agents that speak different languages is advanced and discussed, even if within a relatively related technical setting.

A thorough comparison with language games and game theory is possible only when we would enrich our basic paradigm with an explicit system of “pay-offs,” as to control the agent’s actions and preferences, as well as adding the paradigm with a topological space for both meanings and expressions (respectively structures and basic formulas in our setting). As in language games (but in some contrast to game theory), in our paradigm meaning plays an important role.

We deserve a final remark on game theory and an important objection, namely: “What this model-theoretic approach gives us, over say ‘classical’ game theory and equilibria selection in repeated games (apart being novel and rigorous)?” In fact, Nash equilibrium captures a *steady state* of the play of a strategic game [12], similarly to our notion of stable state of knowledge. On the other hand, it is well known that such and similar notions of equilibrium as proposed by classical game theory [12] are not fully satisfactory in dealing with infinitely repeated games. A thorough comparison with game theory, although very interesting and important, is out of the scope of this paper.

6 CONCLUSION

To summarise, we have advanced and discussed a formal paradigm of coordination, wherein meaning negotiation plays a major role in the process of convergence to a common agreement. We have argued, sometimes by examples taken from application or real world, sometimes by more abstract yet illustrative results (Theorem 1 and Theorem 2), that the paradigm has something perhaps fruitful to offer, namely, a novel and rigorous model-theoretic perspective, based on principles of inductive logic, along the important current debates in Artificial Intelligence and related areas, on multiple agents in interaction, including those on the semantic interoperability, the update semantics and belief revision, and the language creation, evolution and learning. We have called such paradigm and the related perspective: model-coordination.

7 FUTURE WORK

The paradigm can be modified in various ways—we mention one.

Data available to the agents can be limited to just atomic (basic) formulas (no negations), or enriched to include universal or other kinds of formulas. Many alternative definitions of an “environment” as in Definition 1 are possible: partial or incomplete, noisy, imperfect, recursive. We have chosen to consider “complete” environments in this paper as it seems the more simple solution. This is, however,

a questionable representation of many natural environments and the majority of computable environments, where software agents usually interact. It is worth noting that “real” environments may suffer omissions, erroneous intrusions, or both omissions and intrusions. Although we can modify our paradigm to model such environments with some additional effort, these important kind of environments have been outside the scope of this paper. We fix them here for future work on coordination through inductive meaning negotiation.

8 ACKNOWLEDGMENTS

I thank an anonymous reviewer for useful comments. Mr Tonini of ITC-irst provided some problems to solve. Thanks to Paolo Traverso.

REFERENCES

- [1] A. Agostini, *Paradigms of Coordination and Solvability*, Ph.D. dissertation, Università di Siena, 2001.
- [2] L. Amgoud, S. Parsons, and N. Maudet, ‘Arguments, dialogue, and negotiation’, in *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-00)*, ed., W. Horn, pp. 338–342, Amsterdam, (2000). IOS Press.
- [3] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, ‘Semantic schema matching’, in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, eds., R. Meersman and Z. Tari, pp. 347–365, Berlin Heidelberg, (2005). Springer-Verlag LNCS 3760.
- [4] W. Hodges, *Model Theory*, Encyclopedia of mathematics and its applications, v. 42, Cambridge University Press, Cambridge, 1993.
- [5] S. Jain, D. Osherson, J. Royer, and A. Sharma, *Systems That Learn, 2nd edition*, The MIT Press, Cambridge, MA, 1999.
- [6] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, ‘Automated Negotiation: Prospects, methods and challenges’, *International Journal of Group Decision and Negotiation*, **10**(2), 199–215, (2001).
- [7] K. T. Kelly, *The Logic of Reliable Inquiry*, Oxford University Press, New York, NY, 1996.
- [8] C.V. Goldman M. Allen and S. Zilberstein, ‘Language learning in multi-agent systems’, in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 1649–1650, San Mateo, CA, (2005). Morgan Kaufmann.
- [9] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy, ‘Representing and reasoning about mappings between domain models’, in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, eds., R. Dechter, M. Kearns, and R. Sutton, pp. 80–86, Menlo Park, CA, (2002). AAAI Press/The MIT Press.
- [10] E. Martin and D. Osherson, *Elements of Scientific Inquiry*, The MIT Press, Cambridge, MA, 1998.
- [11] A. Omicini, ‘Towards a notion of agent coordination context’, in *Process Coordination and Ubiquitous Computing*, eds., D. Marinescu and C. Lee, chapter 12, 187–200, CRC Press, (2002).
- [12] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, The MIT Press, Cambridge, MA, 1994.
- [13] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, 1994.
- [14] M. Parameswaran, A. Susarla, and A.B. Whinston, ‘P2P networking: An information-sharing alternative’, *IEEE Computer*, **34**(7), 31–38, (2001).
- [15] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, The MIT Press, Cambridge, MA, 1994.
- [16] L. Steels, F. Kaplan, A. McIntyre, and J. Van Looveren, ‘Crucial factors in the origins of word-meanings’, in *The Transition to Language*, ed., A. Wray, 214–217, Oxford University Press, Oxford, UK, (2002).
- [17] K. P. Sycara, ‘Argumentation: Planning other agents’ plans’, in *Proceedings of IJCAI-89*, ed., N. S. Sridharan, pp. 517–523, San Mateo, CA, (1989). Morgan Kaufmann.
- [18] L. Wittgenstein, *Philosophical Investigations*, Blackwell, Oxford, UK, 1953.
- [19] M. Wooldridge and S. Parsons, ‘Languages for negotiation’, in *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-00)*, ed., W. Horn, pp. 393–397, Amsterdam, (2000). IOS Press.

Reaching Agreements for Coalition Formation through Derivation of Agents' Intentions

Samir Aknine¹ and Onn Shehory²

Abstract. This paper addresses the coalition formation problem in multiagent systems. Although several coalition formation models exist today, coalition formation using these models remains costly. As a consequence, applying these models through several iterations when required becomes time-consuming. This paper proposes a new coalition formation mechanism (CFM) to reduce this execution cost. This mechanism is based on four principles: (1) the use of information on task relationships so as to reduce the computational complexity of the coalition formation; (2) the exploitation of the coalition proposals formulated by certain agents in order to derive their intentions, (this principle makes the search for solutions easier, which in turn may result in earlier consensus and agreements-the intention derivation process is performed on a new graph structure introduced in this paper); (3) the use of several strategies for propagating the proposals of the agents in the coalition formation process; and (4) the dynamic reorganization of previous coalitions.

1 INTRODUCTION

The topic of coalition formation has been the subject of much interest over the last few years. Several models have been developed to control the behaviors of the agents involved in the coalition formation process [3,4,5,6]. These models have addressed important issues such as the computational complexity and the use of concessions, to name but two. However, the strategic behaviors of agents during the coalition formation process have not been properly addressed.

This paper tackles this problem and proposes a mechanism based on several strategic behaviors for the agents: (1) concessions as a means of convergence towards the preferred coalitions for the agents with a study of the relationships between tasks in order to facilitate the formation of coalitions according to how compatible the agents' interests are. (2) exploiting the coalition proposals formulated by the agents to derive the underlying intentions. We use these intentions to focus on solutions which lead to earlier consensus. (3) the use of various strategies for propagating the proposals of an agent in order to reinforce its choices. (4) the reorganization of previous coalitions when changes occur in the execution context, or unexpected events tied to the previous coalitions occur.

[1,2] have developed an approach using compromise strategy, but assume that the value for a beneficial compromise is static (fixed or experimentally computed).

In our approach, the aim is to bring agents to identify the compromise point themselves given the context of the coalition formation process they are involved in. Our compromise computation is based on the quid pro quo strategy.

Task relationship analysis can improve coalition search, as we evidenced through the results of our experiments. In particular, the existence of task relationships (e.g. dependencies between tasks) introduces some dependencies between the agents which perform them. When these dependencies exist, it is preferable that agents identify them early on, since such knowledge can promote compromise. Hence, it should be advantageous to identify dependencies between tasks prior to their assignment to agents. Moreover derivation of intentions makes it possible for self-interested agents to adapt their decisions and strategies to the reactions of the other agents while formulating and propagating their proposals for coalitions. A large solution space can make compromises between agents hard to reach, thus making it impossible or quite difficult to explore these solutions with respect to how long the coalition formation process lasts. Therefore agents should inflect their positions with respect to other agents. Our technique for deriving intentions makes it possible to arrive at reciprocal compromises. Based on the principles discussed above we have developed a novel coalition formation mechanism.

The paper is organized as follows. Section 2 introduces the coalition formation problem. Section 3 presents the concepts of the CFM, first the principles of this mechanism for task analysis and the derivation of agents' intentions, then the behaviors of the agents in this mechanism. Section 4 is a brief conclusion.

2 DESCRIPTION AND ILLUSTRATION OF THE PROBLEM

This section defines the coalition formation problem and describes the example we use to illustrate our mechanism.

2.1 Problem Description

Consider a set of agents $N = \{a_1, a_2, \dots, a_n\}$ and a set of tasks $T = \{T_1, T_2, \dots, T_m\}$. The agents in N need to execute the tasks in T and can negotiate their execution. Tasks in T can be combined into sets of tasks, which can be negotiated. Each agent a_i has its own utility function U_i that it tries to maximize. The goal of an agent is to determine the coalitions to join and the task sets to be performed in these coalitions.

¹ LIP6, Université Paris 6, Samir.Aknine@lip6.fr

² IBM Haifa Research Lab, Haifa University, onn@il.ibm.com

2.2 Problem Illustration

In order to illustrate our approach for agent coalition formation and re-formation, let us take the following simple example, to help understand our mechanism (cf. Figure 1).

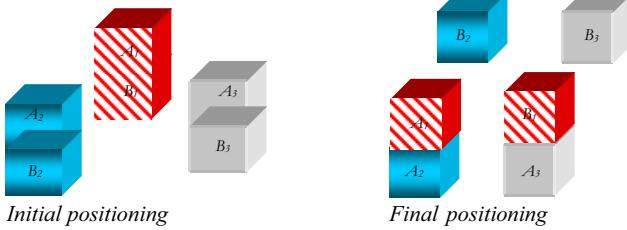


Figure 1. Initial and final block positions

The example concerns agents $a_{i|i=1..n}$ behaving in a world with six blocks initially in a given position. At the end of their execution, agents must have stacked the blocks in a predefined order and at another position (Figure 1). Each agent a_i is responsible for two blocks (A_i, B_i). They have a set of operators for domain actions: $Push(<\{a_i\}>; x; t_k; t_k)$ is the action agent $\{a_i\}$ executes in order to push block x from position t_k to t_k ; $Unstack(<\{a_i\}>; x; y)$ is the action agent $\{a_i\}$ executes in order to unstack block x from y ; $Stack(<\{a_i\}>; x; y)$ is the action agent $\{a_i\}$ executes to stack block x on block y .

3. COALITIONS OF AGENTS

Let us begin by recalling some important definitions.

3.1 Definitions

Definition 1 (Coalition) A tuple $C = \langle A_c, T_c \rangle$ where $A_c = \{a_1, a_2, \dots, a_k\}$ is a set of agents that joined together to perform a task combination $T_c \subseteq T$.

Definition 2 (Coalition structure) A set of coalitions $CS = \{C\}$ such that $\cup A_c \subseteq N$, and $\cup T_c \subseteq T$. A coalition structure which is approved by the agents is considered to be a solution. A solution guarantees that the tasks are executed by the agents. The set $\{T_c\}$ of task combinations that the coalitions in the solution will perform is called the support of the coalition structure.

Definition 3 (Group of coalition structures) A set of coalition structures that, for a specific agent, each provides it with the same utility.

Note that a coalition structure is acceptable for agent a_i if it is preferred over, or equivalent to, the reference structure RS , which corresponds to the minimal guaranteed gain during the negotiation. That is, $U_i(CS) \geq U_i(RS)$.

Definition 4 (Approval set) An approval set $App(CS)$ of a coalition structure CS is the set of agents that have approved the coalition structure CS .

We consider the unacceptable coalition structures as a set of coalition structures $Out(a_i)$ which agent a_i notifies to the others. An unacceptable coalition structure cannot be proposed as a

solution by any agent.

The search for the support of a solution considers relationships between tasks (temporal, causal, etc). Note that complete coverage of the space of task relationship types is beyond the scope of this study. Let us just say that we have to consider task relationships based on the structure of the tasks, then consider relationships based on agents which will perform them. For instance, we have a dependence relationship between $T_1 = Stack(<\{a_1\}>, B_1, A_3)$ and $T_2 = Unstack(<\{a_1\}>, A_1, B_1)$ since the execution of the task T_1 depends on the prior execution of T_2 . Each agent responsible for blocks A_3 and B_1 (respectively a_3 and a_1) is in a relationship (cf. Figure 1). Agent a_1 that is responsible for B_1 may find it very beneficial to satisfy the request of a_3 that is responsible for block A_3 , for instance, for the task $T_3 = Unstack(<\{a_3\}>, A_3, B_3)$, if a_1 wants to reach the requirements for T_1 . However a_1 may also ask for something in exchange, i.e. assistance for $T_4 = stack(<\{a_1\}>, B_1, A_3)$. This cooperation is based on the simple principle of the quid pro quo strategy.

3.2 Coalition formation processing

The coalition formation mechanism requires some processing in addition to the negotiation itself: (1) task analysis prior to negotiation and (2) derivation of agents' intentions during negotiation. The first processing facilitates the concessions on coalitions and the second helps the negotiation converge rapidly.

3.2.1 Task analysis

Task analysis consists of grouping tasks into combinations based on relationships between the tasks, where each combination is to be performed by a coalition. To simplify and optimize the search among those combinations of tasks we construct a binary tree whose nodes are task combinations. The advantage of the suggested method is in directing the search for the support of the solutions towards preferred tasks, thus reducing search complexity.

Definition 5 (Task combination tree) An agent's (binary) task combination tree is built based on the agent's combination preference following a decreasing preference rate for task combinations. Hence, the most preferred combination is the root of the tree. The rest of the nodes are populated in a descending order of preference. Two branches emanate from each node n : an edge labeled (+) and an edge labeled (-). Solutions are placed at the leaves. When inspecting a path from the root to a solution node, (+) on an edge indicates that the node from which the edge emanates is in the support of the solution; (-) indicates that the node from which the edge emanates is not in the support of the solution.

For example, in Figure 3 the support S_3 is only formed from the combinations $\langle T_1, T_2, T_3 \rangle, \langle T_5, T_6 \rangle$ since on the path starting from the root to the leaf S_3 , the node $\langle T_4, T_5 \rangle$ is labeled negatively. Hence, this node does not belong to the support S_3 . The combination tree is built incrementally. To build it, an agent first organizes task combinations in partitions. A partition

is a set of task combinations concerned with one task and containing all the combinations where this task appears. Each task combination belongs to only one partition (cf. Figure 2). For instance, consider a set of agents $\{a_i\}$ and eight tasks T_1, \dots, T_8 . To build its coalition structures, a_i computes some task combinations. This is done with respect to its preferences. For instance, a_i computes the following preferred combinations: $\{T_1, T_2, T_3\}$, $\{T_1, T_2\}$, $\{T_4, T_5\}$, $\{T_5, T_6\}$, $\{T_3, T_4, T_5\}$, $\{T_6, T_7\}$ and $\{T_6, T_7, T_8\}$. To simplify the construction of and the search in the support tree, these combinations are first classified into partitions p_1 , p_2 and p_3 (cf. Figure 2) and ordered in decreasing order of preference. The preference degree of each partition is the average of the preference degrees of the combinations it includes. Here partition p_1 has the highest preference degree and is thus placed at the top.

While building the tree, the preferred combination of agent a_i , $\{T_1, T_2, T_3\}$ is attached to the root of its tree (cf. Figure 3). Then, a_i creates two branches and searches for the next combination to be considered. In the positive branch, the combination must be selected from a partition other than p_1 , because combinations in p_1 share at least T_1 , and a support should have only one instance of each task. The agent selects the combination $\{T_4, T_5\}$ from p_2 . In the negative branch, the agent selects the combination $\{T_1, T_2\}$ from p_1 . The tree is gradually built in the same manner. For instance, the following positive combination is $\{T_6, T_7\}$ from p_3 .

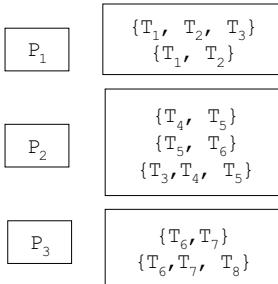


Figure 2. Partitions of task combinations

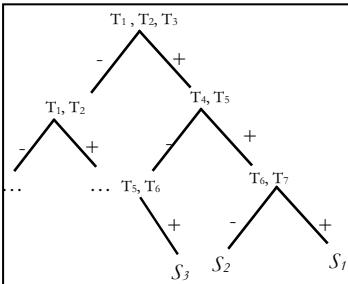


Figure 3. Support tree

Searching the tree and selecting the nodes from which positively labeled branches emanate already gives us several possible supports, S_3 for instance. S_3 is formed from two combinations $\langle\{T_1, T_2, T_3\}, \{T_5, T_6\}\rangle$. In the tree building and search stage each agent autonomously computes its own tree and supports. Each tree may provide several supports of different coalition structures, and supports may vary across trees. This in turn allows several different solutions, as each task combination in each support can be allocated to different coalitions of agents.

3.2.2 Analysis of agent intentions

Each agent applies the same principle of support tree building as well as the generation of the coalition structures to be proposed to the other agents. However, while negotiating, an agent a_i receives several proposals for coalition structures which it must in turn approve, refuse or modify and where appropriate propose new coalition structures to be discussed in later iterations. The construction of these new coalition structures

should not only focus on the choices of a_i but also on the intentions to achieve revealed by the other agents during preceding iterations of the negotiation. This requires a_i to analyze other agents' proposals. To carry out this analysis each agent builds an intention graph relating to the agents which it prefers to approach (possibly all).

3.2.2.1 Building of intention graph

In the previous example, let us assume that agent a_1 received several coalition structures from agent a_2 during the first iterations of the negotiation. Some of the coalition structures are listed below:

$$CS_{21} = \langle \{a_2, a_1\}; \{T_1, T_2, T_3\} \rangle; \langle \{a_2, a_3, a_4\}; \{T_4, T_5\} \rangle;$$

$$\langle \{a_2, a_4\} ; \{T_6\} \rangle$$

$$CS_{22} = \langle \{a_2, a_1\}; \{T_3, T_6\} \rangle; \langle \{a_2, a_3, a_4\}; \{T_2, T_4, T_5\} \rangle$$

$$CS_{23} = \langle \{a_2, a_3\} ; \{T_1, T_2\} \rangle; \langle \{a_2, a_3, a_1\} ; \{T_4, T_5\} \rangle$$

In order to facilitate the convergence of the negotiation, agent a_1 may find it very beneficial to take into account the choices of others, here agent a_2 , in the formulation of the coalition structures. To carry out this processing the agent uses the intention graph to adapt its search for solutions.

Definition 6 (Agent intention graph) An acyclic graph organized in several levels. The nodes of the first level of this graph are the partitions resulting from the classification of the task combinations. The intermediate nodes contain the tasks shared by the nodes they link. The terminal node of the graph is empty.

Using the coalition structure sent by agent a_2 , the agent a_1 builds the set of partitions on the left side of the graph (cf. Figure 4). To build these partitions, a_1 computes the preference of a_2 for each task indicated in the coalition structures. It thus distinguishes the priority and the frequency of a task $\{T_k\}$ in order to compute the preference of a_2 for $\{T_k\}$. The priority of $\{T_k\}$, $P_j(\{T_k\})$, is defined with respect to the group j of coalition structures to which $\{T_k\}$ belongs. A task of the first group of coalition structures obviously has a higher priority than a task appearing in the last group. As for the frequency of $\{T_k\}$ in a group j , $F_j(\{T_k\})$, it depends on the number of occurrences of $\{T_k\}$ in the coalitions structures of this group. The preference of a_2 for $\{T_k\}$, denoted $PREF(\{T_k\})$, is the ratio between its frequency and its priority: $PREF(\{T_k\}) = \sum_{j \geq 1} F_j(\{T_k\}) / P_j(\{T_k\})$.

Based on the preference of each task, a_1 builds the partitions in Figure 4. The first partition p'_1 corresponds to the preferred task T_1 . This partition contains all the combinations where T_1 appears. Therefore, each task combination will belong to only one partition. In each partition described in the left part of Figure 4, a_1 summarizes the combinations of tasks as well as the coalition structures in which they appear. To build the intention graph of agent a_2 , a_1 exploits these task combinations. Using these partitions, a_1 searches first for shared tasks between the combinations in each partition. In Figure 4, note that each partition is attached to a node containing the set of tasks shared by all its combinations. For example, the partition p'_1 , which is initially defined for the task T_1 , is now linked to the node $\langle T_1, T_2 \rangle$. The partition p'_2 , defined for $\langle T_2 \rangle$, is linked to the node $\langle T_2 \rangle$. Once all the nodes of the second level have been built (these nodes indicate the characteristics common to the task combinations of a given partition), a_1 seeks to identify the characteristics common to the partitions, which appear in the

intermediate nodes. To create the nodes of the following level, a_1 analyses the nodes of the current level to check if they share tasks. For each set of common tasks which does not correspond to any node, the agent constructs a new node, which is attached to the nodes which generated it. In the same way, the inclusion of a node in others is also identified in these links, e.g. here the node $\langle T_1, T_2 \rangle$ is attached to the node $\langle T_2 \rangle$ since $\langle T_2 \rangle$ is included in $\langle T_1, T_2 \rangle$.

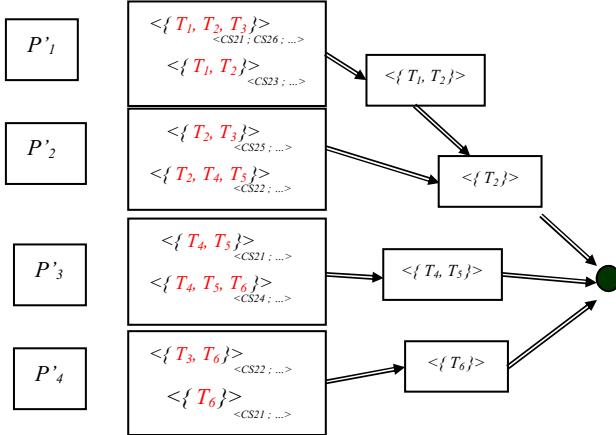


Figure 4. Intention graph for a_2

Note that in this example a_1 builds a graph relating to a_2 , based on the structures that a_2 revealed to a_1 . Depending on the strategy of a_1 , this processing may be repeated: either (1) individually and for all the agents with which a_1 exchanged; or (2) only for the agents which a_1 prefers; or (3) globally by combining in the same intention graph all significant coalition structures for these agents and then computing the preference of these structures. This knowledge is very useful for a_1 since it can use it to deduce the structures these agents prefer to form and accelerate convergence.

3.2.2.2 Intention graph exploration and use

Each agent builds its support tree from which it will propose coalition structures to other agents. These agents will accept or refuse them after negotiation. To build this tree, an agent should also use the knowledge it collected from other agents, i.e. on coalition structures they proposed. These coalition structures and their supports are collected in the intention graph described above. The agent uses this graph in order to find the combinations to insert in its own support tree. While building this tree, each agent analyzes the intention graph it has identified. Remember that each path of the support tree returns a proposal for a coalition structure. Once a task combination has been added to a branch of the support tree, the partition of the intention graph in which this combination appears should be marked with the reference of the branch of the support tree so that it will not be reused in later branches. Thus, in order to avoid the useless exploration of partitions in the intention graph, the agent refers to the following properties:

Property 1 If an ancestor node of a partition p_i in the intention graph contains a task of the path under construction in the support tree, p_i should be marked.

Property 2 Let p_i and p_j be two partitions in the intention graph such that p_i has n_k as a direct ancestor in the intention graph. If p_j is also linked to n_k , marking p_i implies marking p_j too.

To illustrate the use of this intention graph, let's consider again the example presented in section 3.2.1. Recall that agent a_1 computed some task combinations with respect to its preferences: $\{T_1, T_2, T_3\}$, $\{T_1, T_2\}$, $\{T_4, T_5\}$, $\{T_5, T_6\}$, $\{T_3, T_4, T_5\}$, $\{T_6, T_7\}$ and $\{T_6, T_7, T_8\}$. After several rounds of negotiations, agent a_1 received from other agents several proposals which enabled it to build the intention graph shown in Figure 4. To build its first support S_1 , a_1 uses the combinations it has built in its own partitions (cf. Figure 2) as well as the intention graph as explained below.

The first combination of tasks of p_1 is $Comb_1 = \{T_1, T_2, T_3\}$ (cf. Figure 2), which becomes the root of its support tree (cf. Figure 3). a_1 now searches for the combinations which can serve as descendants of $Comb_1$. Of course these combinations should not contain the tasks in $Comb_1$. The exploration of the intention graph is done from right to left and top-down due to the decreasing preferences for the combinations in this graph (cf. section 3.2.2.1). This exploration indicates a first node, $\{T_2\}$ (cf. Figure 4), to which property 1 could be applied. Task T_2 is already in the set of tasks of the branch under construction formed by $Comb_1$. The identification of this node involves marking the first ancestor of this node, i.e. the node $\{T_1, T_2\}$. This ancestor is marked without having to explore it. The first partition p'_1 linked to the node $\{T_1, T_2\}$ is also marked without exploring it (cf. Figure 4). Property 1 implies marking the partition p'_2 of which this node is a direct ancestor. At present, there are only two partitions left in the intention graph which have not been visited (p'_3 and p'_4). a_1 searches for a new combination in order to finish building its branch. It should choose the combination which is the most satisfactory for it and for the other agents. In its partitions, a_1 finds the combination $Comb_2 = \{T_4, T_5\}$ of p_2 (cf. Figure 2). Fortunately in the intention graph the next node to explore is $\{T_4, T_5\}$. This is a favorable situation, since this combination seems to be the best both for a_1 and for the other agents. Again, to finish building its solution, a_1 searches for a combination in the partition p_3 (cf. Figure 2). It has to choose between two combinations and it chooses $\{T_6, T_7\}$. According to the intention graph, this combination also seems to be preferred since the next node to be visited is $\{T_6\}$. The construction of the support tree continues until the agents converge to the same solutions.

3.3 The coalition formation mechanism

The coalition formation mechanism is based on three steps:

(1) The agents initialize the negotiation and transfer their tasks. Each agent asks the other agents to send it their tasks with their explicit priority. Each agent receives the set of tasks and computes the first coalitions to be proposed or agreed to (cf. section 3.2).

(2) An initiator agent computes the first preferred task combinations. For each preferred task combination, the initiator agent finds its preferred agents to perform them. The agent then gathers these coalition structures in groups in order to initiate the negotiation. It then chooses an agent to which these coalition structures will be sent (as proposals for coalition formation). This choice is based on the agent's strategies. The

initiator can behave in three different ways:

- (a) it can send its proposals sequentially to other agents;
- (b) it can send the same proposal to different agents at the same time;
- (c) it can send different proposals to several agents at the same time.

Each strategy involves a particular behavior of the initiator that it should manage. Only strategy (a) is described here. So the initiator initially sends to one agent its preferred coalition structures; it may then send iteratively, in decreasing order of preference, its other coalition structures, until there are no more coalition structures at least equivalent to the reference structure. However, before sending a lower preference coalition structure, the agent may wait until it receives a message from another agent - either a response to its earlier proposal or a new proposal from that agent. These proposals are then used to build the intention graph of these agents (cf. section 3.2.2.1). Each of the other agents also computes its preferred coalition structures. The initiator then computes the second preferred coalition and so on. When an agent r receives a group of coalition structures G from a sender agent s , it only sorts coalition structures that are at least equivalent to the reference structure and the others are not considered. If there is at least one coalition structure CS^* which is preferable or equivalent to r 's best choice, it forwards this CS^* to the next agent that it wishes to include in the negotiation. r approves the CS^* and adds it to $App(CS^*)$. When r finds unacceptable coalition structures in the group it receives, it has to declare them unacceptable to the other agents in $Out(r)$. $Out(r)$ also contains the coalition structures that r itself identified locally as unacceptable. A possible end point of the negotiation occurs when an agent receives a group of coalition structures approved by all the agents concerned in this group. Once the last agent has identified a Pareto optimal solution which is approved by all these agents, it sends them this coalition structure, which they accept as the solution for the negotiation.

(3) The final step of the negotiation consists in validating this solution definitively.

3.4 Coalition formation example

Let us recall the example in section 2.2. Each of a_1 , a_2 and a_3 has identified a set of tasks it has to perform. These tasks are described below.

a_1	a_2	a_3
$T_1 = UnStack (\langle \{a_1, ?\}, A_1, B_1 \rangle)$	$T_6 = UnStack (\langle \{a_2, ?\}, A_2, B_2 \rangle)$	
$T_2 = Stack (\langle \{a_1, ?\}, A_1, A_2 \rangle)$	$T_7 = Push (\langle \{a_2, ?\}, A_2, t_{21}, t_{21} \rangle)$	
$T_3 = Stack (\langle \{a_1, ?\}, B_1, A_3 \rangle)$	$T_8 = Push (\langle \{a_2, ?\}, B_2, t_{22}, t_{22} \rangle)$	
$T_4 = Push (\langle \{a_1, ?\}, A_1, t_{11}, t_{11} \rangle)$	$T_9 = UnStack (\langle \{a_3, ?\}, A_3, B_3 \rangle)$	
$T_5 = Push (\langle \{a_1, ?\}, B_1, t_{12}, t_{12} \rangle)$	$T_{10} = Push (\langle \{a_3, ?\}, A_3, t_{31}, t_{31} \rangle)$	
	$T_{11} = Push (\langle \{a_3, ?\}, B_3, t_{32}, t_{32} \rangle)$	

Once the tasks have been exchanged, each agent tries to find suitable task combinations. Each agent then generates its preferred task combinations and instantiates them. The quid pro quo strategy brings agents a_1 , a_2 and a_3 identify the following combinations after various series of negotiations on feasible combinations: $\{T_1, T_6\}$, $\{T_4, T_7\}$, $\{T_3, T_9\}$, $\{T_2, T_8\}$, $\{T_5, T_{11}\}$.

Consequently agents a_1 , a_2 and a_3 may respectively formulate the following coalition structure proposals:

$$\begin{aligned} SC_1(a_1) &= \langle \{a_1, a_2\} ; \{T_1, T_6\} \rangle ; \langle \{a_1, a_2\} ; \{T_2, T_8\} \rangle ; \langle \{a_1, a_3\} ; \\ &\quad \{T_3, T_9\} \rangle ; \langle \{a_1, a_3\} ; \{T_4, T_7\} \rangle ; \langle \{a_1, a_3\} ; \{T_5, T_{11}\} \rangle ; \langle \dots \rangle \\ SC_2(a_2) &= \langle \{a_1, a_2\} ; \{T_1, T_6\} \rangle ; \langle \{a_1, a_2\} ; \{T_2, T_8\} \rangle ; \langle \{a_1, a_2\} ; \\ &\quad \{T_4, T_7\} \rangle ; \langle \dots \rangle \\ SC_3(a_3) &= \langle \{a_1, a_3\} ; \{T_3, T_9\} \rangle ; \langle \{a_1, a_3\} ; \{T_5, T_{10}, T_{11}\} \rangle ; \langle \dots \rangle \end{aligned}$$

So far each agent has formulated partial solutions considering the tasks it is supposed to carry out. Once the first SC of a_1 is transmitted to a_2 , the latter can immediately approve it since this SC enables it to carry out all its tasks. a_2 may find it beneficial to accept this proposal since it seems completely equitable for it. At this point, only a_1 and a_2 have approved SC_1 . a_1 should now submit a proposal to a_3 . SC_1 is almost satisfactory for a_3 , but unfortunately task T_{10} is not considered in SC_1 . a_3 thus formulates a new coalition structure SC_4 by taking into account the intentions of the agents a_1 and a_2 .

$$SC_4(a_3) = \langle \{a_1, a_2\} ; \{T_1, T_6\} \rangle ; \langle \{a_1, a_2\} ; \{T_2, T_8\} \rangle ; \langle \{a_1, a_3\} ; \\ \{T_3, T_9\} \rangle ; \langle \{a_1, a_3\} ; \{T_4, T_7\} \rangle ; \langle \{a_1, a_3\} ; \{T_5, T_{10}, T_{11}\} \rangle$$

This solution does not weaken the choice of agent a_2 . a_3 can thus consider that it can be approved by a_2 . It only remains for a_1 to approve it, and agree to share task T_{10} with a_3 . The agent that may find it beneficial to yield is a_1 since this solution enables it to carry out all its tasks, unless agent a_2 accepts to carry out T_{10} . SC_4 can thus be approved by all the agents and be used as a solution for the coalition formation.

4 CONCLUSION

This paper has proposed an original coalition formation mechanism, enriched with several essential principles in order to improve the coalition formation: handling relationships between tasks, considering the intentions of the agents during the negotiation of a solution, introducing several strategies for propagating the proposals of the agents in order to reach a solution during the negotiation of the coalitions. This method has been implemented and tested on the application described previously, and compared with the naive method in which these new principles are not studied. The experiments carried out have shown the usefulness of this approach in reducing the time needed for coalition formation and the number of coalition structures evaluated. The approach suggested favors the concessions and allows the agents to converge rapidly towards an acceptable solution.

REFERENCES

- [1] Kraus, S., Shehory, O., Tasse, G. *Coalition Formation with Uncertain Heterogeneous Information*, AAMAS, 1-8, 2003.
- [2] Kraus, S., Shehory, O., Tasse, G., *The Advantages of Compromising in Coalition Formation with Incomplete Information*, AAMAS, 588-595, 2004.
- [3] Rahwan, T., Jennings, N. R. *Distributing coalitional value calculation among cooperating agents*, AAAI, 152-157, 2005.
- [4] Sandholm T.W., Lesser V.R. *Coalitions among Computationally Bounded Agents*, AI, V94-1, 99-137, 1997.
- [5] Sandholm T.W., Larson K., Andersson M., Shehory O., Tohmé F. *Coalition Structure Generation with Worst Case Guarantees*, Artificial Intelligence, V111, 209-238, 1999.
- [6] Shehory O., Kraus S. *Methods for Task Allocation via Agent Coalition Formation*, AI, V101, 165-200, 1998.

Cheating is not playing: *Methodological Issues of Computational Game Theory*

Bruno Beaufils¹ and Philippe Mathieu²

Abstract. Computational Game Theory is a way to study and evaluate behaviors using game theory models, via agent-based computer simulations. One of the most known example of this approach is the famous Classical Iterated Prisoner's Dilemma (CIPD). It has been popularized by Axelrod in the beginning of the eighties and had led him to set up a successful Theory of Cooperation.

This use of simulations has always been a challenging application of computer science, and of agent-based approaches, in particular to Social Sciences. It may be viewed as *Empirical Game Theory*. These kind of approach is often necessary since, in the general case, classical analytical ones do not give suitable results. These tools are also often used when full game-theoretic analysis is intractable.

The usual method to evaluate behaviors consists in the collection of strategies, through open contests, and the confrontation of all of them as in a sport championship. Then it becomes, or at least seems to become, easy to evaluate and compare the efficiency of these behaviors.

Evaluating strategies can however not be done efficiently without the insurance that algorithms used are well formed and that they can not introduce bias in their computation. It can not be done without tools able to prevent or, at least, measure deviation from the object of the study. Unfortunately people using such simulations often do not take care seriously about all those aspects, because they are not aware of it, and sometimes because they are. We will try to show effects of bad simulations practice on the simplest example.

We show methodological issues which have to be taken care of, or avoided in order to prevent trouble in simulation results interpretation. Based on some simple illustration, we exhibit two kinds of bias that could be introduced. We classify them as voluntary or involuntary mistakes. The former ones can be explained by poor design of experimentations whereas the latter can defeat the purpose of the evaluation using simple ideas of agreement and cooperation. We also show the implications on interpretations and conclusions that such errors may produce.

We state that scoring/ranking methods are part of the game, and as such have to be described with the game. Many points described may seem to be widely known. We think that with the growth of interest of such methods they have to be detailed and exposed clearly.

1 Introduction

Social sciences generally deal with the study of human beings behavior and their effect on the behavior of groups of humans beings. Studies in this field use two methodologies. The first one consists,

as in *natural science*, in the observation of real life situations in order to try to determine which laws govern the observed groups. The second one, whose adoption is much more recent, consists in the construction of formal models which are then tested and which results are compared to observations. Actually, those two methodologies are not as distinct as one may think. Often the first method is used as first step for the second one in a infinite loop trying to understand the world.

A lot of scientific fields are using this kind of methods to achieve their goal. One may think to economic and management science, trying to understand the behavior of group of people, psychology and artificial intelligence, trying to understand or mimic the behavior of individual. Some mathematical tool have been set up with such a goal in mind. Game Theory has, for instance, been mainly set up in order to understand and explain the behavior of people in parlor game, [8]. It has been used in many political situation, trying to solve conflicts (as this year Economics Nobel Prize winner work) and is still very largely used to understand economic behavior.

With the growth of computer power the testing steps of these approaches are more frequent. The main reason is that purely mathematical tools are sometimes nor powerful nor usable enough when dealing with individual elements and not sets. These tools are very well adapted for a macroscopic view of the world but lose a lot of attractiveness when trying to have a view at a microscopic level. These inappropriate perspectives may be caused by the classical continuous approach of mathematical tool used to explain a world which is a discrete one. On the other side, computers and agent-based models deal very well with discrete computations. Combination of discrete mathematical models and computation power of today is a real chance for *social science* at large.

Another explanation of this proliferation of computer simulations use may be that people think that computer are easier to use than mathematics. The problem is that it is generally not true. We will try to show difficulties involved when using computer agent-based simulations along with some basic solutions which are usable.

We will focus on one particular example which is the study of individual behavior and their effect mainly on the individual fitness but also on the group behavior. This is exactly what computational game theory or *empirical game theory* ([9]), is about. While classical game theory try to solve (or at least to explain) conflictual situations and evolutionary game theory try to explain population dynamics, computational game theory is about evaluating individual behavior. Within this context the purpose of the work is then to find some good behavior, also called strategies. What *good* really means depends on the macroscopic effect studied (cooperation, coalition dynamic, etc.).

The classical way to do is to simulate a big tournament implying a lot of different behavior in a specific game. The more efficient the

¹ LIFL, University of Sciences and Technologies of Lille, beaufils@lifl.fr

² LIFL, University of Sciences and Technologies of Lille, mathieu@lifl.fr

behavior is in this game the better it is considered. Diversity in nature of behavior present in the population of agents implied is then very important. Evaluating an agent in a population where every individuals behave the same way is not interesting, at least most of the time. In order to ensure this needed diversity scientists often use a way which seems efficient : ask other scientists how they will behave in the specified game. That has been successful more than once. One of the main reason is that every participant has not the same representation of the situation modeled by the game. Since background of participants are different, they even do not have the same idea of how to deal well in such situation.

Evaluating strategies can not be done efficiently without testing it more than once. It can not be done without the insurance that algorithms used are well formed and that they can not introduce bias in the result of simulations. It can not be done without tools able to prevent, or at least to measure, deviation from the object of the study.

Unfortunately people using computer simulations does not always take care seriously about all these aspects (CEC'04 contest), because they are not aware of it, and sometimes because they are. We will try to show effects of bad simulations practice on the simplest example.

In a first section we will describe computational game theory methods which will be used to illustrate our ideas. A specific example will be chosen, described, and used in all the rest of the paper. The next section will describe some bias that could be introduced by those methods when they are not used in a proper manner or simply because they are not well designed. That could be summarized as *how to cheat without communication*. Then the third section will described how strategies involved can defeat the purpose of the evaluation using trivial ideas based on agreement and cooperation. That will be summarized as *how to cheat with communication*.

2 Computational Game Theory

Game Theory may be seen as a mathematical tool designed to understand what is a rational behavior in specific situations. More precisely behavior are called strategies. Strategies describe in detail what to do in any possible situation. Possible situations are described by some rules. A definitive definition of what Game Theory is about is still very discussed and two different approaches still coexist. Even the *real life* applicability of Game Theory is still widely discussed. One can refer, among others, to [3, chapter 1] or [7, chapter 5].

We would like to show that evaluating strategies in game theory by computer simulations has to be done with scientific rigor and needs some methodological precautions. One has also to be aware of limits implied by those issues. We focus the demonstration on iterated and repeated games involving two players. The main used evaluation method is based on sport championship. The idea is to evaluate and then order some strategies in a specific game. Evaluation is done pairwise, for every possible pair. In this paper we concentrate, as an example, on the iterated prisoner's dilemma, see [6] for complete history and informal presentation. Remarks made in this context stay however true on any iterated game.

Classically, these kind of evaluations are done during a specific contest where any kind of people are allowed to submit strategies, through a simple description or a more detailed source code implementation in a fixed programming language. The main benefit of using such situation is clarity, since distinction between people being evaluated (*participants*) and people evaluating (*referees*) are obvious.

Two cases are then observable: (i) implementation mistake or design flaws (ii) organized cheating. In both cases we will show that it is almost always possible to get a well-evaluated strategy at the end

of the contest although it is actually a very poor one. In the first case it is often due to a naive software implementation implying edge effect. Referee, and not participants, has then to be blamed. It could be defined as *involuntary cheating*. In the second case an unfair participant is able to profit, for instance, of a design flaw in the contest organization, in order to favor a strategy it has submitted. It could be defined as *voluntary cheating*.

2.1 Iterated games

Iterated games studied here are based on the repetition of a simple game which is defined by a single payoff matrix.

Every iteration of a game is defined by choices made by players. Choices consist of a selection in a fixed list of available moves³. Combination of the move selected by players in a specific iteration defines an outcome of the simple game. The payoff matrix defines payoff obtained by each player for any reachable outcome in an iteration. Iteration will be later called round; the complete sequence of iterations referenced as a meeting, a match or simply a game. The final payoff of a player is simply the sum of all iteration's payoff. The goal of all players is the same: to maximize their final payoff.

A player is identified by predetermined choices following a strategy. This strategy allows, for any given iteration, to know the move a player will use according to previous reached outcomes and in particular according to previously moves played by opponents.

Using computer science terminology a strategy is identified to a program and a behavior during an iterate game to an execution of this program.

In the CIPD there are only two players and two moves available in the simple game. These moves are noted C (for Cooperation) and D (for Defection). For every iteration players choose their moves simultaneously. The payoff matrix is presented in table 1. This matrix is known by both players.

Table 1. Classical Iterated Prisoner's Dilemma payoff matrix.

	C	D
C	3	0
D	5	1

This matrix is a Prisoner's Dilemma payoff matrix. This game involves 2 players and 2 different moves (C and D). Available moves are the same for each player. It is a symmetric matrix: payoff of player A for outcome (C,D), that is when A played C and B played D, is the same as payoff for player B for the outcome (D,C), that is when A played D and B played C; which is 0. Matrix shows thus only the payoff of row player.

In this paper we will only use simple strategies usable in the CIPD:

- a11_c always plays C whatever opponent played.
- a11_d always plays D whatever opponent played.
- tit_for_tat plays C as first move, then plays whatever move opponent played in the previous iteration.
- spiteful plays C until the opponent played D, then always D.
- soft_majo plays the move the opponent has used more often in previous iterations (the majority move). If it played C as many time as D then plays C.
- per_X plays periodically moves in the sequence X. For instance per_CDD, also noted as (CDD)*, plays periodically C,D,D.

³ These moves are the available strategies of the simple game

2.2 Evaluation method

The most used strategy's evaluation method is simply the comparison of its payoff to the one of another strategy. The higher the payoff is, the better the strategy is considered. To evaluate all strategies of a fixed set it is then sufficient to sum scores of each strategies against every other and finally to rank them on the basis of their scores.

For a two-player game, computing such a *round-robin* tournament on set S containing n strategies is simply to fill a $n \times n$ matrix and then to sum value of each lines in order to get the score of each strategy.

It has to be noticed that in such tournament a strategy's score does not necessarily include its score when playing against itself. To the end of the paper and in order not to deal explicitly with the contribution of such inclusion, at least in term of robustness, we will consider that all cells on the diagonal of tournament matrices are nulled. That simplification does not change anything to ideas developed herein and almost nothing to examples chosen as illustrations.

With such a method the higher a strategy is ranked the better its value is considered. Quality of the evaluation depends however essentially in the size of the strategies set.

Classically in order to get a big number of strategies one may organize a contest open to everyone. It is asked to every participant to submit **one** strategy which respects rules of the played game. A big tournament is then computed with all submitted strategies as the set to be evaluated. The basic idea is here to allow the research of new *good* strategies for a specific game. The contest is used as a way to insure diversity and heterogeneity in behaviors.

This method has been regularly used, in particular on the CIPD. One can think to Robert Axelrod contest in the beginning of eighties, see [1], to the French edition of the Scientific American in the beginning of nineties, see [4] for the announcement and [5] for the results, as well as to some scientific conferences such as Artificial Life V in 1996 or the Conference on Evolutionary Computation in 2000. In scientific conferences the goal is often to test new techniques (such as evolutionary computation) when applied to strategies for the iterated prisoner's dilemma.

In all cases *modus operandi* is always the same. Someone asks people to submit strategies. Let us call it the organizer. It has the responsibility to collect strategies and to test them in the most objective way using computer simulations. People submitting strategies are called participants. They submit their proposition by sending the organizer some source code of a computer program, a textual description, or simply by filling a web form, explaining the behavior they want to propose. The organizer generally is in charge of objectivity of the contest. It is the one who runs simulations and thoroughly study their results. The only purpose of such contest is to determine some efficient strategies in a specific situation, not to find a *realistic* mathematical model of a social interactive situation. The organizer insures that all participants are aware of that fact.

3 Cheating without communicating

3.1 Normalization

One of the well known problems with iterated version of prisoner's dilemma is the length of games. If both players know when the meeting will end (last iteration) then an iterated game is equivalent to a simple one. In such situation both players have to play D all the time. It has to be avoided.

One of the chosen solution is to use a discount rate. It allows to give less signification to future moves compared to already played

one. For any formal studies with infinite meeting without any real or computed simulations this make sense. Outside of this context it becomes very difficult to chose a value for this rate and even more difficult to justify this choice. Moreover infinite is not a realizable concept in computer science.

When using simulations it is often preferred to pseudo randomly choose a length at the beginning of every game of a tournament. This pseudo-random number generation is done for a fixed variance. Result of the generation is of course not available to players nor before, nor during the meeting.

It may however be possible for an unfortunate player to always get short meeting compared to the average length. In such conditions it may be badly evaluated whereas, with some luck, evaluation could be completely different. Let us illustrate this using 3 strategies `all_d`, `tit_for_tat`, $(CD)^*$, and consider the following game length:

Players	Game Length
$(CD)^*$ vs <code>all_d</code>	m
<code>all_d</code> vs <code>tit_for_tat</code>	n
<code>tit_for_tat</code> vs $(CD)^*$	p

According to these length, to the CIPD payoff matrix (table 1) and the definition of strategies, payoff obtained by each strategies during a tournament are:

	<code>all_d</code>	$(CD)^*$	<code>tit_for_tat</code>
<code>all_d</code>		$6(m/2)$	$n + 4$
$(CD)^*$	$m/2$		$(5p/2) + 3$
<code>tit_for_tat</code>	$n - 1$	$(5p/2) - 2$	

With $n = 50$, $m = 100$, $p = 60$, which means an average deviation of 20, the obtained rank is:

1. `all_d` with 354 points ;
2. $(CD)^*$ with 203 points ;
3. `tit_for_tat` with 197 points.

On the other hand with $n = 50$, $m = 60$, $p = 100$, which means we stay at a average deviation of 20, the rank becomes:

1. `tit_for_tat` with 297 points ;
2. $(CD)^*$ with 283 points ;
3. `all_d` with 234 points.

It is important to notice that these 2 ranks have completed different head and tail. It is also to be noticed that average deviation and standard deviation stay constant in both cases: 20 and 26.46.

With a normalization of game length to $n = m = p$, we then get the following rank:

1. `all_d` with $4(n + 1)$ points ;
2. $(CD)^*$ with $3(n + 1)$ points ;
3. `tit_for_tat` with $\frac{7n}{2} - 3$ points.

Naturally for the two previous tournament using another normalization process, consisting in the computation of the average score for each iteration, we get a similar rank, which is however different from the two firsts:

1. `all_d` with an average payoff of 4 points ;
2. `tit_for_tat` with an average payoff of 3 points ;
3. $(CD)^*$ with an average payoff of 3.5 points.

Game length has to be unknown by players. Contest organizer has however to choose between using the same pseudo-randomly computed length for every game in the contest, or normalizing scores according to number of rounds played by each strategies. If a tournament is used as an evaluation method player do not have to play against itself.

In the rest of this paper we will use game of 10 rounds for every example and illustration.

3.2 Clones strategies

In some particular context 2 different strategies (program) may produce the same behavior (execution). Let us consider for illustration `tit_for_tat` and `spiteful`. They are different since they use different concepts. They however produce the same behavior against 2 strategies as uncommon as `all_c` and `all_d`.

In a general manner the same case may arise when dealing with genotype and phenotype. Using computer science terminology it is trivial to say that two completely different programs may produce the same results.

That may be viewed from another angle. Many strategies with different names may finally be exactly the same ones. In such case, the use of such a strategy in a contest is a clear bias of the final evaluation and thus devalue the quality of it.

If, for instance, we use n times the same strategy `S1` (by means of same code but different names) in a set of strategies all different when compared by pair (by mean of different code), then it may be possible that `S1` is the best evaluated one. When the same strategy is used only once with the same set of competitors then it may be another strategy than `S1` which is considered the best.

Considering `tit_for_tat` as `S1` as well as `all_d`, and `(CD)*` for the set of competitors it is then easy to see that such situation may arise very clearly, even with tournament composed by game of only 10 rounds:

TOURNAMENT RANK	TOURNAMENT RANK
1 : <code>all_d</code> = 44	1 : <code>tit_for_tat</code> = 62
2 : <code>per_cd</code> = 33	2 : <code>tit_for_tat</code> = 62
3 : <code>tit_for_tat</code> = 32	3 : <code>per_cd</code> = 61
	4 : <code>all_d</code> = 58

We showed how the repeated use of a strategy may influence results of tournament. This frequent bias may arise in 2 cases:

- when 2 strategies have similar implementation code but different names. It is then very easy for a participant of a contest to favor its proposition by flooding it with a very big number of the same strategy using different names each time. In another context (combinatorial auctions, see [10]) a problem close to this one has been identified as *false-name bid*.
- when 2 strategies have different code, different names but produce the same execution due to the simulation context. It is a classical problem in computer science since there is an infinite number of way to write the same thing.

Moreover, it has to be noticed that deciding if two strategies are similar in the general case is impossible. That is a very strong knowledge in theoretical computer science derived from the fact that it is impossible to decide if two programs (Turing machines) have identical behavior (accept the same language). This decision problem is not recursively enumerable. It is however perfectly decidable in some more constraints cases where strategies are defined by filling a predefined structure. This last method is often used on web contest where

participants fill web form in order to define a strategy. This is, however, less interesting since, in such cases, it is often possible to make some tournament involving all possible instances, see work presented in [2] as an example.

4 Cheating with communication

In most cases, players do not know strategies used by their opponents. They may try to deduce it from the behavior of others players. This is a difficulty and one of the major interest of using computational game theory, which we can call *behavioral inference*. At the opposite, let us suppose that you know the strategy of your opponent before playing. In this case it could be easy to determine the best reply to it and thus to maximize your payoff.

Although it is forbidden to communicate *directly* with its opponent, in order for instance to make a deal and agree on the behavior to adopt, it is possible to use the history of played moves to communicate *indirectly*. One of the most trivial idea is then to use some kind of starter succession of moves, used by others to identify you.

Communication can then be established by some coding system fixed in advance by the strategy creator.

With such hypothesis two kind of cases may be considered:

- a unique strategy used by different participants who try to recognize themselves;
- different strategies whose goal of some is to favor the profit of some other.

4.1 Strategies which recognize themselves

We illustrate one such case through a strategy which plays a specific starter succession of moves and then cooperates always after this succession if it observed the same starter played by the opponent and else defects.

If half the population uses the same strategy, then one member of the subpopulation using it has a great chance to be the winner. On the other hand if only one participant uses it in the population then it has a great chance of being a loser since it could no more take advantage from the communication.

Let us consider the `cheater` strategy which plays the `(CDDC)` starter, then observe what the opponent played in the 4 first moves. If opponent played the same succession of moves as the starter then `cheater` cooperates always, else it defects always.

If 3 players using `cheater` are tested against `tit_for_tat`, `spiteful` and `soft_majo` the tournament winner is `cheater`. If the same experiment is done with only one instance of `cheater` then it appears to be the last of the final tournament rank:

TOURNAMENT RANK	TOURNAMENT RANK
1 : <code>cheater2</code> = 109	1 : <code>spiteful</code> = 75
2 : <code>cheater3</code> = 109	2 : <code>tit_for_tat</code> = 74
3 : <code>cheater</code> = 109	3 : <code>soft_majo</code> = 73
4 : <code>spiteful</code> = 105	4 : <code>cheater</code> = 57
5 : <code>tit_for_tat</code> = 102	
6 : <code>soft_majo</code> = 99	

Once 2 players using `cheater` recognizes themselves they win 3 points each and on each round, whereas the rest of time they make others players losing at least 2 points.

Generally this kind of behavior is easily identifiable on tournament results since all strategy using it are grouped at the same level in the final rank (at least when using purely deterministic strategies). This

does not show the same effect as in the previous section where the same strategy is used by more than one player. Here strategies have different behavior adapted to different opponent.

4.2 Master/Slaves

On the same principle as in the previous example it is possible to establish a strategy taking advantage of some others agreeing strategies. The one who benefits is said to be the master and the others to be the slaves.

Compared to the previous example once strategies have recognized themselves then on each round the master win 5 points defecting against its slaves, which agree to be slave by cooperating only with the master. Against others players slaves may use any other aggressive behavior.

Let us consider that the master strategy plays (CDDC), then if it is recognized the same starter played by its opponent always plays D, and if not plays as tit_for_tat.

The slave plays exactly as cheater: it plays (CDDC) then if the opponent did the same starter it cooperates, and else defects.

If master is evaluated against cheater, tit_for_tat and soft_majo it wins the tournament whereas the slave one is ranked last. In the case of no slave the master takes the last position:

TOURNAMENT RANK		TOURNAMENT RANK	
1 :	master = 105	1 :	tit_for_tat = 84
2 :	tit_for_tat = 98	2 :	soft_majo = 83
3 :	soft_majo = 96	3 :	spiteful = 75
4 :	spiteful = 90	4 :	master = 67
5 :	cheater = 65		

Such example could even be refined considering one master but n slaves with completely different starter for each slave. The master job would then only be to recognize the different slaves and to use some other rather *good* strategies against *real* opponent such as tit_for_tat or spiteful.

It has to be noticed that the harmful effect of slaves can be selective. For instance a slight modification of cheater (replacing all_d by tit_for_tat) may harm only aggressive strategies.

In a real contest one has however to find another participant who accepts to be slave, that is using a strategy which have a lot of chance to be badly evaluated. If such coalitions could be identified one solution is then to set the score of each member of the coalition as the average payoff of all member of it. Unfortunately this identification is often difficult to do

Public contest may avoid as far as possible the possibility for one participant to offer more than one strategy. Collusion, such as those shown here, may nevertheless appear since agreement or coalition between different participants before the contest are not avoidable.

The final rank has then to be particularly well studied in all its details, and, at least, strategy of players with the same final payoff have to be investigated.

5 Conclusion

Computational Game Theory deals with the evaluation of heuristic strategies using iterated game. It is some kind of empirical Game Theory. When full game-theoretic analysis is intractable or unuseful, and, as in the general case, it is not possible to prove formally that a strategy is better than another, computer simulations become essential. It generally uses a very specific multi-agent based approach

where agent's behaviors are defined by human participants in public contest. All around the world this method is very often used on different kind of game.

Through the CIPD, we showed in this paper that this kind of evaluation can not be done without any precaution, and that false, or at least biased, results may be found. It is very easy to favor one particular kind of strategy, consciously or not. We described different types of skews one should be aware of and take care about (computation method, repetition of strategies, master/slave effects, etc.).

All along the presentation we tried to give number of examples which we described precisely so that they can be verified and reproduced. We tried to make a short survey of what has to be done or not when using computer simulations for iterated game strategies comparison. Even if you could not present them here, methodological solutions to problem presented here exist. They are mainly based on evolutionary computation ideas and on sets of reference.

In this particular context of strategies evaluation, we basically state that scoring/ranking methods are part of the game. They thus have to be described with the game. All participants must have the same level of information on the contest in order, for instance, to arm themselves against some flaws described here. Changing the purpose of a contest after its start or favoring some participants is cheating.

Many points described here may seem to be widely known or at least trivial. With the rise of contests falling in all traps described here, we fear that finally they are not so well known. We think that with the growth of interest of such methods they have to be detailed and at least exposed clearly.

Computational, as Empirical Game Theory are crossing a major step of its evolution. It may be a very efficient method for evaluating and later constructing new *artificial intelligent* behaviors. In order to achieve this goal, it has to be used seriously.

6 Acknowledgements

Tools used to made experiments presented here are all available on the web at the following address: <http://www.lifl.fr/IPD>.

Works presented here has been supported by the Nord/Pas-de-Calais regional council (through CPER TAC project) and the European Regional Development Fund. Authors would like to thank all of theses institution for their support.

REFERENCES

- [1] R. Axelrod, *The evolution of cooperation*, Basic Books, New-York, USA, 1984.
- [2] B. Beauflis, J.-P. Delahaye, and P. Mathieu, ‘Complete classes of strategies for the classical iterated prisoner’s dilemma’, volume 1447 of *Lecture Notes in Computer Science*, 33–41, Springer-Verlag, (1998).
- [3] K. Binmore, *Essays on the Foundations of Game Theory*, Blackwell, 1990.
- [4] J.-P. Delahaye, ‘L’altruisme récompensé’, *Pour La Science (French Edition of Scientific American)*, **181**, 150–156, (1992).
- [5] J.-P. Delahaye and P. Mathieu, ‘L’altruisme perfectionné’, *Pour La Science (French Edition of Scientific American)*, **187**, 102–107, (1993).
- [6] W. Poundstone, *Prisoner’s Dilemma*, Doubleday, 1992.
- [7] A. Rubinstein, *Economics and Language*, Cambridge University Press, 2000.
- [8] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944. The standard reference is the revised edition of 1947.
- [9] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart, ‘Analyzing complex strategic interactions in multi-agent games’, in *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, (2002).
- [10] M. Yokoo, Y. Sakurai, and S. Matsubara, ‘The effect of false-name bids in combinatorial auctions: new fraud in internet auctions’, *Games and Economic Behavior*, **46**, 174–188, (2004).

Mediation in the framework of morpho-logic

Isabelle Bloch¹ and Ramón Pino-Pérez² and Carlos Uzcátegui²

Abstract. In this paper we introduce a median operator between two sets of interpretations (worlds) in a finite propositional language. Our definition is based on morphological operations and Hausdorff distance. It provides a result which lies “halfway” between both sets and accounts for the “extension” or “shape” of the sets. We prove several interesting properties of this operator and compare it with fusion operators. This new operator allows performing mediation between two sets of beliefs, preferences, demands, in an original way, showing an interesting behavior that was not possible to achieve using existing operators.

1 INTRODUCTION

Consider the situation of two agents who are settling the differences between the demands (goals, etc.) of the members of the parties they represent. One of the negotiating conditions is that each demand should receive a fair treatment. The final output of the process should be a new set of demands (for instance, for the merging of the two parties).

This problem reminds the problem faced by fusion [3] where a finite collection of belief sets is merged into another belief set. However, a general fusion operator will not take into account each belief of each party, only consensual beliefs are important. For instance, let X and Y represent respectively the demands of the parties. When X and Y have non empty intersection, any fusion operator will let $X \cap Y$ to be the result of the fusion. The reason is that fusion relies mostly on the parts of X and Y which are the “closest” to each other. The purpose of this paper is to introduce a *median operator* \odot such that $X \odot Y$ lies “halfway” between X and Y and partly satisfies the demands of every member of both parties. This will be achieved by means of the Hausdorff distance which takes into account the “extension” or “shape” of X and Y . The idea of the median is coming from works in Mathematical Morphology about interpolations between compact sets in metric spaces [6]. The idea of Morpho-logic has been already useful for dealing with others operators that naturally appear in AI problems [2].

As it happens with fusion operators, where beliefs are given at the semantical level, we assume that the sets of demands X and Y are given as sets of interpretations (worlds) in a finite propositional language. We use a distance function d between worlds to measure the similitude between worlds (for all practical purposes the reader can assume that d is the Hamming distance). The distance d gives a method to represent the closest worlds to a set X . In fact, let $\delta(X)$ be the collection of all worlds w such that $d(w, X) \leq 1$, where $d(w, X) = \min\{d(w, x) : x \in X\}$. In morphological terms, $\delta(X)$

is called a dilation of X by a ball of radius 1. More generally, the dilation of X of any radius r is the set $\delta_r(X)$ of all w such that $d(w, X) \leq r$.

The Hausdorff distance between X and Y is the smallest integer ρ such that $X \subseteq \delta_\rho(Y)$ and $Y \subseteq \delta_\rho(X)$. Then ρ measures the compromise that both X and Y should make in order to satisfy all demands of the other party. Our median operator satisfies that $d(z, X \odot Y) \leq \rho/2$ for all $z \in X \cup Y$. It is in this sense that $X \odot Y$ lies “halfway” between X and Y . This is a consequence of the fact that \odot has a stronger property: if $x \in X$ and $y \in Y$ then any path from x to y of length at most ρ will meet $X \odot Y$ (the details are given later).

We first recall, in Section 2, the definition of some fusion operators. In Section 3 we introduce a median operator constructed via dilation and Hausdorff distance, study some of its properties and compare it with the fusion operator. In Section 4 we introduce another median operator constructed via dilation, erosion and Hausdorff distance; we compare this operator to the previous one and study some of its properties. In Section 5 we present a procedural approach to compute \odot .

2 FUSION

In this section we recall some previous definitions of fusion [2]. In order to make the exposition more readable, we assume that Ω , the set of worlds, is finite and we fix the distance d as the Hamming distance between worlds. Nevertheless our results hold for all distances $d : \Omega^2 \rightarrow \mathbb{N}$ having the following property (of normality): for all $X \subseteq \Omega$ such that $X \neq \emptyset$ and $X \neq \Omega$, there is a $\omega \in \Omega \setminus X$ such that $d(\omega, X) = 1$.

Let us begin recalling the morphological definition [2] of the Δ^{max} operator [3], denoted here by Δ in order to simplify the notation:

$$X\Delta Y = \delta_n(X) \cap \delta_n(Y) \quad (1)$$

where $n = \min\{k : \delta_k(X) \cap \delta_k(Y) \neq \emptyset\}$.

Since dilations are defined using centered balls of a distance d as structuring elements, they are extensive, i.e. $X \subseteq \delta_r(X)$ (this property will be often used in the following). Other useful properties of dilation are iterativity ($\delta_r(\delta_{r'}(X)) = \delta_{r+r'}(X)$) and monotony ($X \subseteq Y \Rightarrow \delta_r(X) \subseteq \delta_r(Y)$).

Let $m = d_{\min}(X, Y)$ be the minimum distance between X and Y , i.e. $d_{\min}(X, Y) = \min\{d(x, y) : x \in X, y \in Y\}$. Our next definition is a variant of Equation (1):

$$X\Delta' Y = \begin{cases} \delta_{m/2}(X) \cap \delta_{m/2}(Y) & \text{if } m \text{ is even,} \\ (\delta_{\lfloor m/2 \rfloor}(X) \cap \delta_{\lfloor m/2 \rfloor+1}(Y)) \\ \cup (\delta_{\lfloor m/2 \rfloor+1}(X) \cap \delta_{\lfloor m/2 \rfloor}(Y)) & \text{if } m \text{ is odd.} \end{cases} \quad (2)$$

¹ GET - Ecole Nationale Supérieure des Télécommunications - CNRS UMR 5141 LTCI, 46 rue Barrault, 75013 Paris, France - Isabelle.Bloch@enst.fr

² Departamento de Matemáticas - Facultad de Ciencias - Universidad de Los Andes, Mérida, Venezuela - pino@ula.ve, uzca@ula.ve

Note that if m is even, Δ' and Δ are identical operators. In case m is odd, this new definition avoids dilating “too much”. Then Δ' may be preferred to Δ .

In general, $X\Delta'Y \neq X\Delta Y$ if m is odd (in this case, the value of n in equation (1) is $n = \lfloor m/2 \rfloor + 1$). In general we have:

$$X\Delta'Y \subseteq X\Delta Y \quad (3)$$

(with an equality if m is even). This is illustrated in Figure 1.

Note that $X\Delta Y = X \cap Y$ if $X \cap Y \neq \emptyset$ (in this case $n = 0$).

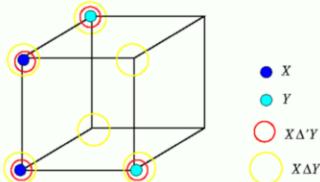


Figure 1. Fusion using Δ and Δ' . The vertices of the cube represent worlds and edges link worlds at distance 1 from each other.

There are some relationships between the two well known fusion operators Δ^{\max} and Δ^Σ ³ [3]. We have, in fact, the following proposition:

Proposition 1 For any X and Y , $X\Delta'Y = \Delta^{\max}(X, Y) \cap \Delta^\Sigma(X, Y)$.

It has been proved in [4] that Δ^Σ is a merging operator and Δ^{\max} is a quasi-merging operator. Thus Δ' satisfies all the common postulates (of merging and quasi-merging) that pass to the intersection. For instance, the consistency axiom: $X \cap Y \neq \emptyset$ entails $X\Delta'Y = X \cap Y$.

Next, we introduce some definitions which will be useful in the following. Namely, they will be helpful to show that the operator Δ' can be characterized as in Proposition 2.

A path α between two elements x and y of Ω is a sequence, x_0, \dots, x_n of points in Ω such that $d(x_i, x_{i+1}) = 1$ for any $i = 0, \dots, n - 1$. The length of such a path α , denoted $|\alpha|$, is n . The set of paths between x and y is denoted $P(x, y)$. If $\alpha = x_0, \dots, x_n$ is a path, we define the set of central points of α , denoted $c(\alpha)$, as $\{x_{n/2}\}$ if n is even and $\{x_{\frac{n-1}{2}}, x_{\frac{n+1}{2}}\}$ otherwise. We will say that an operator Δ_1 is *smaller or equal* than an operator Δ_2 if and only if for any input (X, Y) , $X\Delta_1 Y \subseteq X\Delta_2 Y$.

Proposition 2 The operator Δ' is the smallest operator \diamond having the following property: for all $x \in X$, for all $y \in Y$ and for all path $\alpha \in P(x, y)$, if $|\alpha| \leq m$ then $c(\alpha) \subseteq X \diamond Y$, where $m = d_{\min}(X, Y)$.

Even if this operator enjoys good properties, it is not adequate for some purposes, namely in order to get a global consensus as the following simple example shows:

Example 3 Suppose X and Y represent the political agenda of the Labor Party and the Green Party, respectively. They would like to find an agreement in order to present a common platform for the next elections. As it usually happens, there are several leanings in each of the two parties. In order to simplify, suppose there are two leanings in each party, the left and central. Their positions about ten parameters are encoded by worlds. The parameters are the following: nationalization of big private companies; increasing the taxes

³ $\Delta^\Sigma(X, Y) = \{w : d(w, X) + d(w, Y) = \min_{w' \in \Omega} d(w', X) + d(w', Y)\}$

for the rich people; reducing the taxes for the poor people; increasing the budget of public health; increasing the budget for public education; creating a subvention for the unemployed persons; increasing the budget of research programs; reducing the subsidies for the big agriculture exploitations; creating subsidies for organic agriculture; fostering the public transports. The encoding is made in the following manner: $X = \{w_1, w_2\}$, $Y = \{w_1, w_3\}$, where $w_1 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ is the encoding of the left wing in both parties, the central wing in the Labor Party is encoded by $w_2 = (0, 0, 0, 1, 1, 1, 0, 0, 0)$ and the central wing in the Green Party by $w_3 = (0, 0, 0, 0, 0, 0, 1, 1, 1)$. If the policy adopted for merging these leanings is a classical fusion, in which the result is $\{w_1\}$ (the intersection of both groups), they run the risk of the splitting of both political parties. Thus they have to adopt a different policy in order to reach an agreement more fair to all leanings.

The rest of the paper is dedicated to study some operators proposed as an alternative to classical merging operators. These operators are called median operators and take inspiration from Mathematical Morphology [5], in particular [6].

3 MEDIAN FROM HAUSDORFF DISTANCE

One of the main features of the morphological setting is that it offers different ways of talking about closeness. One of them has already been evoked in the previous section when we defined the minimum distance m between X and Y . Actually this notion does not define a distance⁴ between elements of $\mathcal{P}^*(\Omega)$, the set of nonempty subsets of Ω (because $m = 0$ iff $X \cap Y \neq \emptyset$). However, there is a true distance between elements of $\mathcal{P}^*(\Omega)$. This distance, d_H , is the so called Hausdorff distance, defined from d in the following way:

$$d_H(X, Y) = \max\{\max\{d(x, Y) : x \in X\}, \max\{d(y, X) : y \in Y\}\} \quad (4)$$

where $d(w, Z) = \min\{d(w, z) : z \in Z\}$, for $w \in \Omega$, $Z \in \mathcal{P}^*(\Omega)$.

It is easy to check that d_H is indeed a distance. In order to simplify the notation, when X and Y are fixed, we put $\rho = d_H(X, Y)$.

Note that, by definition, ρ can be viewed as a measure of the degree of global disagreement between X and Y : among the distances between an element of X and Y and between an element of Y and X , ρ is the largest one. Another way to understand the meaning of ρ is given by the following identity:

$$\rho = \min\{n : X \subseteq \delta_n(Y) \wedge Y \subseteq \delta_n(X)\} \quad (5)$$

Thus, the set $\delta_\rho(X) \cup \delta_\rho(Y)$ has a natural “fairness property”, but it is in general too big. We propose a better solution guided by the notion of interpolation.

The notion of interpolating set in our discrete case is:

$$Z_n = \delta_n(X) \cap \delta_{\rho-n}(Y) \quad (6)$$

for $n = 0, 1, \dots, \rho$. When n is roughly $\rho/2$, then the interpolating set Z_n “lies half way between” X and Y . Thus, the definition of our median operator is as follows:

$$X \odot Y = \begin{cases} \delta_{\rho/2}(X) \cap \delta_{\rho/2}(Y) & \text{if } \rho \text{ is even,} \\ (\delta_{\lfloor \rho/2 \rfloor}(X) \cap \delta_{\lfloor \rho/2 \rfloor+1}(Y)) \cup (\delta_{\lfloor \rho/2 \rfloor+1}(X) \cap \delta_{\lfloor \rho/2 \rfloor}(Y)) & \text{if } \rho \text{ is odd.} \end{cases} \quad (7)$$

Note that $\lfloor \rho/2 \rfloor + 1 = \rho - \lfloor \rho/2 \rfloor$.

We will show next that \odot is indeed a well defined operator and then show some of its properties.

⁴ Remember that a distance have to satisfy $d(X, Y) = 0 \Leftrightarrow X = Y$; $d(X, Y) = d(Y, X)$ and triangle inequality.

Properties

1. Let $Z_{\rho'} = \delta_{\rho'}(X) \cap \delta_{\rho-\rho'}(Y)$ for $\rho' \in \{0, \dots, \rho\}$. Then $Z_{\rho'} \neq \emptyset$.

Proof: since the Hausdorff distance is ρ , there exists x in X and y in Y and a path $x_0 = x, \dots, x_\rho = y$ of length ρ such that $d(x_i, x_{i+1}) = 1$. For $i = \rho'$, we have $x_{\rho'} \in \delta_{\rho'}(X)$. Since $d(x_{\rho'}, y) \leq \rho - \rho'$, we have also $x_{\rho'} \in \delta_{\rho-\rho'}(Y)$. (The path can be decomposed into a path $x = x_0 \dots x_{\rho'}$ of length ρ' and a path $x_{\rho'} \dots x_\rho = y$ of length $\rho - \rho'$). Hence $x_{\rho'} \in Z_{\rho'}$. ■

This results proves that the definition of $X \odot Y$ is consistent:

$$X \odot Y \neq \emptyset \quad (8)$$

2. Since the dilations are extensive, we have:

$$X \cap Y \subseteq X \odot Y \quad (9)$$

3. Link between median and fusion:

- if $m < \rho$, then $X \Delta Y \subseteq X \odot Y$ (see Figure 2);
- if $m = \rho$ and is an even number, then $X \Delta Y = X \odot Y$ (see Figure 3);
- if $m = \rho = 1$, then $X \odot Y = X \cup Y \subseteq X \Delta Y$.
- if $m = \rho$ and is an odd number, then $X \odot Y = X \Delta' Y$ and therefore (by Proposition 1) $X \odot Y \subseteq X \Delta Y$.

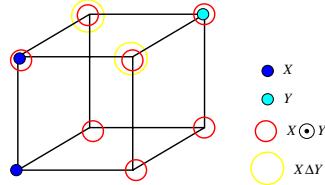


Figure 2. An example where $\rho = 3$ and $m = 2$ and $X \Delta Y \subseteq X \odot Y$.

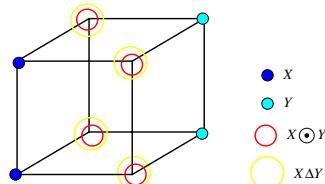


Figure 3. An example where $\rho = m = 2$ and $X \Delta Y = X \odot Y$.

Proof:

- $m < \rho$, ρ and m even: then $n = m/2$, and $X \Delta Y = \delta_{m/2}(X) \cap \delta_{m/2}(Y)$, which is included in $\delta_{\rho/2}(X) \cap \delta_{\rho/2}(Y)$ since $m < \rho$ and dilation is increasing with respect to the size of the structuring element.

- $m < \rho$, ρ odd and m even: then $m < \rho - 1$ hence $2n \leq \rho - 1$ and $n \leq \lfloor \rho/2 \rfloor$. Therefore we have $\delta_n(X) \subseteq \delta_{\lfloor \rho/2 \rfloor}(X) \subseteq \delta_{\lfloor \rho/2 \rfloor + 1}(X)$ and the same for Y , from which we deduce $\delta_n(X) \cap \delta_n(Y) \subseteq X \odot Y$.

- $m < \rho$, m odd: then $n = \lfloor m/2 \rfloor + 1$ ($2n = m + 1$), hence $m + 1 \leq \rho$ and $2n \leq \rho$. If ρ is even, then $n \leq \rho/2$ and $X \Delta Y \subseteq X \odot Y$ (as for the first case). If ρ is odd, since $2n$ is even, we have $2n \leq \rho - 1$ and $n \leq \lfloor \rho/2 \rfloor$. Therefore $X \Delta Y \subseteq X \odot Y$ (as for the second case).

- $m = \rho$ and even: then $n = m/2 = \rho/2$ and $\delta_n(X) \cap \delta_n(Y) = \delta_{\rho/2}(X) \cap \delta_{\rho/2}(Y)$.

- $m = \rho = 1$: then $n = 1$ and $\lfloor \rho/2 \rfloor = 0$. Therefore we have $X \subseteq \delta_1(Y)$ and $Y \subseteq \delta_1(X)$ (from the definition of Hausdorff distance) and $X \odot Y = X \cup Y$. Moreover, $X \cup Y \subseteq \delta_1(X)$ and $X \cup Y \subseteq \delta_1(Y)$, so $X \cup Y \subseteq \delta_1(X) \cap \delta_1(Y) = X \Delta Y$. ■

The last case is illustrated in Figures 4 and 5 in two different situations.

A consequence of this result is that if ω is in $X \Delta Y$ and if $d_{\min}(X, Y) = m$, this does not imply that there is x in X and y in Y such that $d(x, \omega) + d(\omega, y) = m$. An example of ω is shown in Figure 5 for which $d(x, \omega) + d(\omega, y) \geq 2$.

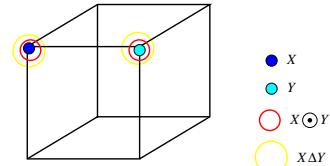


Figure 4. An example where $\rho = m = 1$ and $X \Delta Y = X \odot Y$.

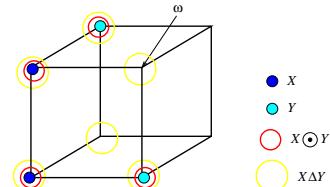


Figure 5. An example where $\rho = m = 1$ and $X \odot Y \subset X \Delta Y$.

Note that this property tells that when $m < \rho$, then $X \Delta Y \subseteq X \odot Y$ and when $m = \rho$, then $X \odot Y \subseteq X \Delta Y$.

4. Commutativity: $X \odot Y = Y \odot X$ (by construction).
5. Since $d_H(X, X) = 0$, we have $X \odot X = X = X \Delta X$.
6. \odot is not associative. A counter-example is shown in Figure 6.

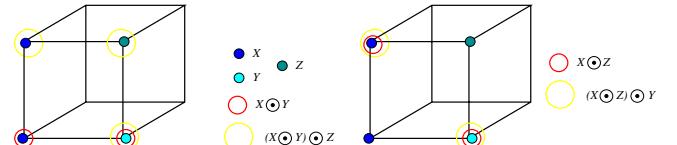


Figure 6. An counter-example of associativity.

7. \odot is not monotone, i.e. $X \subseteq X'$ does not entail generally $X \odot Y \subseteq X' \odot Y$.

Example 3 (continued) In this example we have $\rho = 7$. The point $w = (0, 0, 0, 1, 1, 1, 1, 1, 1) \in X \odot Y$ and it is a distance 3 from w_2 and a distance 4 from w_3 . Notice that, unlike the Δ operator, \odot does not forget w_2 and w_3 . In fact, for every $z \in X \cup Y$, we can find a point $t \in X \odot Y$ such that $d(z, t) \leq 3$: for $z = w_1$ take $t = w_1$; for $z = w_2$ take $t = w$; for $z = w_3$ take $t = (0, 0, 0, 1, 1, 1, 1, 1, 1)$.

Now we present the definition of a median operator. It is motivated by the properties of \odot . We will show that this notion captures the idea of getting a set that lies halfway between X and Y (see proposition 6).

Definition 4 An operator $\diamond : \mathcal{P}^*(\Omega) \times \mathcal{P}^*(\Omega) \rightarrow \mathcal{P}^*(\Omega)$ is said to be a median operator if \diamond satisfies the following condition:

$$\forall x \in X, \forall y \in Y, \forall \alpha \in P(x, y) \quad (|\alpha| \leq \rho \Rightarrow c(\alpha) \subseteq X \diamond Y) \quad (10)$$

Theorem 5 \odot is the smallest median operator, i.e. \odot satisfies Equation (10) and for all $X, Y \in \mathcal{P}^*(\Omega)$ and all operator \diamond satisfying Equation (10), $X \odot Y \subseteq X \diamond Y$.

Proof: It is enough to see that $z \in X \odot Y$ iff $\exists x \in X, \exists y \in Y$ and $\exists \alpha \in P(x, y)$ such that $|\alpha| \leq \rho$ and $z \in c(\alpha)$. The *if* part is easy to show: suppose $\alpha = x_0, x_1, \dots, x_n$ and n is even. Since $z = x_{\frac{n}{2}}$, z is in $\delta_{\frac{\rho}{2}}(X) \cap \delta_{\frac{\rho}{2}}(Y)$ and since $n \leq \rho$ and dilations are increasing we have $z \in \delta_{\frac{\rho}{2}}(X) \cap \delta_{\frac{\rho}{2}}(Y)$ (in case ρ is even) or $z \in (\delta_{\lfloor \rho/2 \rfloor}(X) \cap \delta_{\lfloor \rho/2 \rfloor+1}(Y)) \cup (\delta_{\lfloor \rho/2 \rfloor+1}(X) \cap \delta_{\lfloor \rho/2 \rfloor}(Y))$ (in case ρ is odd). A similar argument works when n is odd.

For the *only if* part, suppose ρ is even and that $z \in \delta_{\frac{\rho}{2}}(X) \cap \delta_{\frac{\rho}{2}}(Y)$. Thus there are two paths $x_0, x_1, \dots, x_k = z$ and $z = y_0, y_1, \dots, y_p$ with $x_0 \in X, y_p \in Y, k \leq \frac{\rho}{2}$ and $p \leq \frac{\rho}{2}$. So $\alpha = x_0, x_1, \dots, x_k, y_1, \dots, y_p$ is a path of length $k + p$. Note that $k + p \leq \rho$. We consider two cases. First, if $|k - p| \leq 1$, $z \in c(\alpha)$. Second, if $|k - p| \geq 2$ the path can be extended, via loops, to a path α' with $|\alpha'| \leq \rho$, $z \in c(\alpha')$ and $\alpha' \in P(x_0, y_p)$ (for instance if $k = p - 2$, $\alpha' = x_0, x_1, x_0, x_1, \dots, x_k, y_1, \dots, y_p$). The case ρ odd is similar. ■

Proposition 6 Suppose \diamond satisfies Equation (10), then the following holds:

$$d(z, X \diamond Y) \leq \rho/2, \text{ for all } z \in X \cup Y.$$

In particular, this is true for \odot . Note that the operator Δ' (which satisfies Proposition 2) does not enjoy a property analogous to Proposition 6, namely, there are X and Y and $z \in X \cup Y$ such that $d(z, X \Delta' Y) > m/2$.

4 MEDIAN FROM DILATION AND EROSION

In this section we present a different median operator. We first recall the definition of the erosion $\varepsilon_k(X)$ of size k of a set X : $w \in \varepsilon_k(X)$ if $\delta_k(w) \subseteq X$.

The idea for the new median operator is to use any parameter $\nu \geq m$ in Equation (7) instead of ρ . This method provides a whole family of operators \odot_ν as follows:

$$X \odot_\nu Y = \begin{cases} \delta_{\nu/2}(X) \cap \delta_{\nu/2}(Y) & \text{if } \nu \text{ is even,} \\ (\delta_{\lfloor \nu/2 \rfloor}(X) \cap \delta_{\lfloor \nu/2 \rfloor+1}(Y)) \\ \cup (\delta_{\lfloor \nu/2 \rfloor+1}(X) \cap \delta_{\lfloor \nu/2 \rfloor}(Y)) & \text{if } \nu \text{ is odd.} \end{cases} \quad (11)$$

Note that if $\nu > \rho$, then \odot_ν is a median operator. For instance, when $\nu = 2\rho$, then $X \cup Y \subseteq \delta_\rho(X) \cap \delta_\rho(Y)$. But the dilations can be very large, and therefore an erosion is applied on the result, leading to the following definition:

$$M'(X, Y) = \varepsilon_k(\delta_\rho(X) \cap \delta_\rho(Y)) \quad (12)$$

This definition is particularly interesting if we take:

$$k = \begin{cases} \rho/2 & \text{if } \rho \text{ is even,} \\ \lfloor \rho/2 \rfloor & \text{if } \rho \text{ is odd.} \end{cases} \quad (13)$$

From the next result and Theorem 2 it follows that M' is indeed a median operator, i.e. it satisfies the condition (10).

Theorem 7

$$X \odot Y \subseteq M'(X, Y) \quad (14)$$

Moreover, if $m < \rho$, then $X \Delta Y \subseteq X \odot Y \subseteq M'(X, Y)$.

Proof:

- ρ even, $k = \rho/2$:

$$X \odot Y \subseteq \varepsilon_{\rho/2}\delta_{\rho/2}(X \odot Y)$$

since $\varepsilon_{\rho/2}\delta_{\rho/2}$ is a closing, hence extensive. Since $\delta(A \cap B) \subseteq \delta(A) \cap \delta(B)$ for any dilation, we have:

$$X \odot Y \subseteq \varepsilon_{\rho/2}(\delta_{\rho/2}\delta_{\rho/2}(X) \cap \delta_{\rho/2}\delta_{\rho/2}(Y))$$

Since $\delta_{\rho/2}\delta_{\rho/2} = \delta_\rho$, we have $X \odot Y \subseteq M'(X, Y)$.

- ρ odd, $k = \lfloor \rho/2 \rfloor$:

$$X \odot Y \subseteq \varepsilon_{\lfloor \rho/2 \rfloor}\delta_{\lfloor \rho/2 \rfloor}(X \odot Y)$$

Since δ commutes with \cup , we have:

$$\begin{aligned} \varepsilon_{\lfloor \rho/2 \rfloor}\delta_{\lfloor \rho/2 \rfloor}(X \odot Y) &= \varepsilon_{\lfloor \rho/2 \rfloor}(\delta_{\lfloor \rho/2 \rfloor}(\delta_{\lfloor \rho/2 \rfloor}(X) \cap \delta_{\lfloor \rho/2 \rfloor+1}(Y)) \\ &\quad \cup \delta_{\lfloor \rho/2 \rfloor}(\delta_{\lfloor \rho/2 \rfloor+1}(X), \delta_{\lfloor \rho/2 \rfloor}(Y))) \end{aligned}$$

Therefore (with $\lfloor \rho/2 \rfloor + \lfloor \rho/2 \rfloor = \rho - 1$):

$$\begin{aligned} X \odot Y &\subseteq \varepsilon_{\lfloor \rho/2 \rfloor}((\delta_{\rho-1}(X) \cap \delta_\rho(Y)) \cup (\delta_\rho(X) \cap \delta_{\rho-1}(Y))) \\ &\subseteq \varepsilon_{\lfloor \rho/2 \rfloor}(\delta_\rho(X) \cap \delta_\rho(Y)) \end{aligned}$$

Indeed:

$$\begin{aligned} (\delta_{\rho-1}(X) \cap \delta_\rho(Y)) \cup (\delta_\rho(X) \cap \delta_{\rho-1}(Y)) &= \\ (\delta_{\rho-1}(X) \cup \delta_\rho(X)) \cap (\delta_{\rho-1}(Y) \cup \delta_\rho(Y)) \cap \dots &= \\ \delta_\rho(X) \cap \delta_\rho(Y) \cap \dots & \end{aligned}$$

which is included in $\delta_\rho(X) \cap \delta_\rho(Y)$. ■

Several examples are shown in Figure 7.

5 ORDERINGS AND PROCEDURAL APPROACH

A procedural approach can be derived from the following table ($\Sigma = d(\omega, X) + d(\omega, Y)$ and $\text{Max} = \max\{d(\omega, X), d(\omega, Y)\}$):

	X	Y	Σ	Max
ω_1	$d(\omega_1, X)$	$d(\omega_1, Y)$	Σ_1	Max_1
ω_2	$d(\omega_2, X)$	$d(\omega_2, Y)$	Σ_2	Max_2
...				
ω_n	$d(\omega_n, X)$	$d(\omega_n, Y)$	Σ_n	Max_n

The procedures for different operators are as follows:

- Δ -fusion (Δ^{max}): take ω_i having the smallest Max_i . This corresponds to the following total pre-order: $(d_1, d_2) \leq (d'_1, d'_2) \Leftrightarrow \max\{d_1, d_2\} \leq \max\{d'_1, d'_2\}$.
- Σ -fusion (Δ^Σ): take ω_i having the smallest Σ_i . This corresponds to the following total pre-order: $(d_1, d_2) \leq (d'_1, d'_2) \Leftrightarrow d_1 + d_2 \leq d'_1 + d'_2$.
- $\Delta' \subseteq \Delta^\Sigma \cap \Delta^{max}$ corresponds to the following order: $(d_1, d_2) \leq (d'_1, d'_2) \Leftrightarrow d_1 \leq d'_1$ and $d_2 \leq d'_2$. Δ' can be obtained from Σ -fusion and a second filter: $|d(\omega, X) - d(\omega, Y)| \leq 1$ (or minimizing the Max_i).

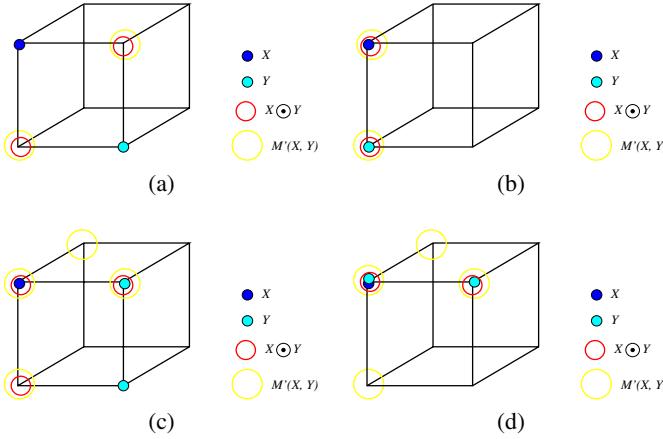


Figure 7. (a) An example with $\rho = m = 2$ and $M'(X, Y) = X \odot Y = X\Delta Y$. (b) An example with $\rho = m = 1$ ($k = 0$) and $M'(X, Y) = X \odot Y = X\Delta Y = X \cup Y$. (c) An example with $\rho = 2$, $m = 1$ ($k = 1$) and $X \odot Y = X\Delta Y$, $X \odot Y \subset M'(X, Y)$. (d) An example with $\rho = 1$, $m = 0$ ($k = 0$), $X\Delta Y = X$, $X \odot Y = X \cup Y$, and $X\Delta Y \subset X \odot Y \subset M'(X, Y)$.

- \odot can be obtained from the table in the following way: first take the set of ω such that $\Sigma \leq \rho$ then among these worlds take those satisfying $|d(\omega, X) - d(\omega, Y)| \leq 1$.
- Note that ρ can be calculated using the table: ρ is the maximum in the column of *Max* for the rows corresponding to worlds of $X \cup Y$.

One of the main features of this procedural approach is that it suggests very precise manners to extend the median operators to more than two elements.

Extensions to more than two elements $X_1 \dots X_k$: We propose three possible extensions:

1. Based on Proposition 1, we extend Δ' using the following procedure: minimize the column of Σ and then minimize the *Max*.
2. ρ can be generalized as follows:

$$\rho = \max_i \{d(\omega, X_i) : \omega \in X_1 \cup \dots \cup X_k\} \quad (15)$$

This number corresponds to the maximum of the columns of *Max* among the rows coming from worlds in $X_1 \cup \dots \cup X_k$.

Let $\bar{\rho} = (k-1)\rho$. First, take all ω such that $\Sigma \leq \bar{\rho}$. For the second step we have two alternatives: (i) keep only those ω such that the sum of $|d(\omega, X_i) - d(\omega, X_j)|$ is minimal; (ii) keep only those ω such that $|d(\omega, X_i) - d(\omega, X_j)| \leq k-1$.

Examples are shown in Figures 8 and 9.

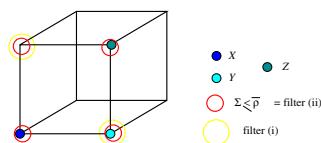


Figure 8. An example of extension to three elements.

3. First compute:

$$A_{ij} = X_i \odot_{2\rho} X_j = \delta_\rho(X_i) \cap \delta_\rho(X_j) \quad (16)$$

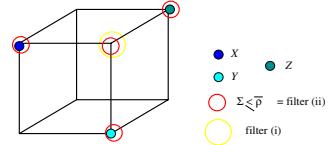


Figure 9. Another example of extension to three elements.

where ρ is given by Equation 15. The proposed operator is $\bigcap_{i \neq j} A_{ij}$. It is easy to show that this intersection is non empty. Notice also that if we take $X_i \odot_\rho X_j$ in (16) the intersection could be empty. Since the intersection could be very large, we can then take the most central part (the last non empty erosion). This is similar to the idea of barycenter. An example is shown in Figure 10.

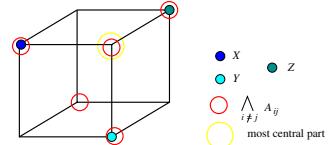


Figure 10. Extension to three elements using the last approach.

6 CONCLUSION

We have proposed the notion of median operator. The minimal of all median operators is denoted by \odot . The operator \odot is useful for computing the “middle interpolators” between two sets of beliefs, preferences or demands. This provides an original way to perform mediation or negotiation, showing interesting results which were not possible to achieve using existing operators. In particular we have shown that all parts of the two input sets are taken into account in the final decision, as opposed to classical fusion operators which account only for the parts that are the closest to each other.

Future work aims at further exploring the proposed extensions to more than two sets. Other definitions could also be developed, based on different morphological approaches, such as influence zones [1, 7]. Further applications to negotiations problems are also foreseen.

REFERENCES

- [1] S. Beucher, ‘Sets, Partitions and Functions Interpolations’, in *Mathematical Morphology and its Applications to Image Processing, ISMM’98*, eds., H. Heijmans and J. Roerdink, pp. 307–314, Amsterdam, The Netherlands, (1998). Kluwer Academic Publishers.
- [2] I. Bloch and J. Lang, ‘Towards Mathematical Morpho-Logics’, in *8th International Conference on Information Processing and Management of Uncertainty in Knowledge based Systems IPMU 2000*, volume III, pp. 1405–1412, Madrid, Spain, (2000).
- [3] S. Konieczny and R. Pino-Pérez, ‘On the logic of merging’, in *KR’98: Principles of Knowledge Representation and Reasoning*, eds., Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, pp. 488–498, San Francisco, California, (1998). Morgan Kaufmann.
- [4] S. Konieczny and R. Pino-Pérez, ‘Merging information under constraints: a logical framework’, *J. Logic Comput.*, **12**(5), 773–808, (2002).
- [5] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New-York, 1982.
- [6] J. Serra, ‘Hausdorff distances and interpolations’, in *Mathematical Morphology and its Applications to Image and Signal Processing*, eds., Henk J. A. M. Heijmans and Jos B. T. M. Roerdink, 107–114, Kluwer Academic Publishers, (1998).
- [7] J. Vidal, J. Crespo, and V. Maojo, ‘Recursive Interpolation Technique for Binary Images based on Morphological Median Sets’, in *Mathematical Morphology and its Applications to Image Processing, ISMM’05*, eds., C. Ronse, L. Najman, and E. Decencière, volume 30, pp. 53–62, Paris, France, (2005). Springer-Verlag.

Strengthening Admissible Coalitions

Guido Boella and Luigi Sauro¹ and Leendert van der Torre²

Abstract. We develop a criterion for coalition formation among goal-directed agents, the indecomposable do-ut-des property. The indecomposable do-ut-des property refines the do-ut-des property (literally *give to get*) by considering the fact that agents prefer to form coalitions whose components cannot be formed independently. A formal description of this property is provided as well as an analysis of algorithms and their complexity.

1 Introduction

In this work we study the formation of coalitions among goal-directed agents. We start from a theory of social power and dependence introduced by Castelfranchi et al. [3]. In this context a coalition is intended as a group of agents which agree to cooperate for the achievement of a shared goal or to exchange with each other the achievement of their own goals. The second case is of particular interest as different networks of exchanges can be considered but not all of them are realizable.

We assume that the formation of a coalition is supported by unanimous and enforced agreements, i.e. a coalition is effectively formed only when all its members agree to it (unanimousness) and they cannot deviate from what established in the agreement, once they decide to enter it (enforcement). Under these assumptions, we develop a criterion of admissibility, the indecomposable do-ut-des property (i-dud in the following), establishing which coalitions cannot be formed under the assumption that the agents are self-interested. The i-dud property is a refinement of the do-ut-des property [2] which describes a condition of reciprocity: an agent *gives* a goal only if this fact enables it to *obtain*, directly or indirectly, the satisfaction of one of its own goals. The i-dud property refines the do-ut-des property by taking into account also the fact that a coalition formation process can itself be costly and usually the costs involved in a coalition formation process increase with the number of agents involved. Furthermore, being a coalition agreed unanimously, the more agents are involved in it, the larger is the risk of defections which can jeopardize the formation of the coalition. Thus, agents prefer to form coalitions which are as small as possible.

In Section 2 we define the i-dud property and provide some examples of this notion. In Section 3, we provide an algorithm to search all the sub-coalitions of a given coalition which satisfies the i-dud property. Even if this problem is not computationally tractable, we show in Section 4 that the problem to verify if a single coalition satisfies the i-dud property is tractable and, in several cases, also the complexity of the first problem may decrease considerably. Conclusions end the paper.

¹ Università di Torino, Italy, email: {guido,sauro}@di.unito.it

² University of Luxembourg, Luxembourg, email: leon.vandertorre@uni.lu

2 From do-ut-des to i-dud coalitions

Depending on the problem, a multiagent system can be represented at different levels of abstraction. For example, if you want to study how agents have to coordinate in order to achieve a goal or how agents should optimally use their resources, then a multiagent system can involve resources, actions, plans [1, 6]. Instead, if you want to study which coalitions are strategically admissible to be formed, you usually do not need such fine-grained descriptions of a multiagent system, coalitions are directly described by means of the consequences that they can attain collaborating, without any description about which joint plans agents have to perform [7, 10]. Following this idea, we describe potential coalitions abstracting from actions, plans or resources. However, in contrast with these approaches, we do not represent a coalition just indicating the set of goals that it can attain. We want to use the topology of goal exchanges inside a coalition to define our admissibility criterion.

Thus, inspired by Conte and Sichman [4] we represent a (potential) coalition as a labeled AND-graph of dependencies among agents. A labeled AND-graph consists of a set of nodes \mathcal{V} - which denotes the agents involved in the coalition - and a set \mathcal{E} of labeled AND-arcs. Denoting with Gl the goals exchanged in the coalition, a labeled AND-arc connects an agent ag_i to a nonempty set of agents Q and it is labelled with a goal $g \in Gl$, so it can be represented as a triple (ag_i, Q, g) . The meaning of such an arc is that the agent ag_i desires the goal g and the achievement of g is delegated to the set of agents Q . In order to represent a coalition a labeled AND-graph has to satisfy two further conditions. Since a coalition is intended as the result of an agreement process, the first condition is that only those agents that contribute to the achievement of some goals are admitted in this process. The second condition establishes that a coalition formation process does not involve private commitments that do not require any form of collaboration.

Definition 1 A coalition is represented as a labeled AND-graph which is a tuple $C = \langle \mathcal{V}, \mathcal{E} \rangle$, where \mathcal{V} is a finite set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times (2^{\mathcal{V}} \setminus \{\emptyset\}) \times Gl$ is a set of labeled AND-arcs.

C satisfies two conditions: (1) for each node $ag_j \in \mathcal{V}$, there exists at least an AND-arc (ag_i, Q, g) such that $ag_j \in Q$ and (2) \mathcal{E} does not contain an AND-arc in the form $(ag_i, \{ag_i\}, g)$.

With an abuse of notation we mean with $(Q, g) \in C$ that there exists a $(ag_i, Q, g) \in \mathcal{E}$. Following [8] we call (Q, g) a commitment of C . A sub-coalition C' is intended a subgraph of C where some commitments are suppressed, $(Q, g) \notin C'$, for some $(Q, g) \in C$.

We want to restrict the notion of coalitions assuring, first, reciprocity, and, second, that no sub-coalitions of a coalition can be formed independently. The former property is called do-ut-des [2, 8], and we refine it to match the second requirement called indecomposability.

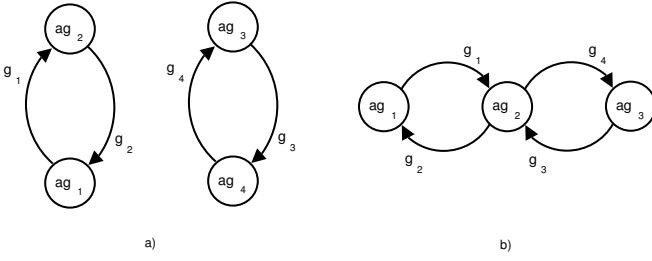


Figure 1. Two coalitions that satisfy the do-ut-des property but that do not satisfy the i-dud property.

The do-ut-des property assures that each chain of exchanges involved in a coalition returns something back to each agent involved in it. This property has been characterized in [2, 8] by means of a qualitative preference relation and a notion of dominance similar to those used in Game Theory, such as the notion of core. Here, instead, we propose a version of the do-ut-des property which is grounded on topological property of chains of exchanges; the equivalence with the dominance-based approach is shown in [8]. In [8] a goal is meant as a state of affairs that, once attained by a group of agents, benefits one or, in case, more agents. To simplify our formalism, we consider that there do not exist two agents which desire the same goal.

We introduce some preliminary notions. A finite sequence of AND-arcs $\mathcal{P} = (ag_{i_1}, Q_1, g_1), \dots, (ag_{i_n}, Q_n, g_n)$ is a path if for all $2 \leq h \leq n$, $ag_{i_h} \in Q_{h-1}$. Paths formalize possible *chains of exchanges* among the agents. We denote with $out(ag_i)$ the set of labeled AND-arcs outgoing the agent ag_i , i.e. $out(ag_i) = \{(ag_j, Q, g)\}$. Given $\mathcal{O} \subseteq out(ag_i)$, we call \mathcal{O}^* the propagation of \mathcal{O} , i.e. the set of AND-arcs containing all the paths $\mathcal{P} = (ag_{i_1}, Q_1, g_1), \dots, (ag_{i_n}, Q_n, g_n)$ such that (1) $(ag_{i_1}, Q_1, g_1) \in \mathcal{O}$ and \mathcal{P} does not contain an arc in $out(ag_i) \setminus \mathcal{O}$. We notice that $out^*(ag_i)$ contains all (and only) the paths starting from ag_i .

The do-ut-des property consists of two conditions. The first condition is an efficiency condition: there do not exist two distinct sets of agents in a coalition which are committed to the achievement of the same goal. The second condition expresses the notion of reciprocity: given an AND-arc $(ag_i, Q, g) \in \mathcal{E}$, each agent ag_j involved in Q agrees to provide the goal g to ag_i , only in the case this commitment returns the satisfaction of some of its goals by means of a path \mathcal{P} .

Definition 2 A coalition $\langle \mathcal{V}, \mathcal{E} \rangle$ satisfies the do-ut-des property iff (1) there do not exist two commitments $(Q, g), (Q', g) \in \mathcal{C}$ such that $Q \neq Q'$ and (2) for all $(ag_i, Q, g) \in \mathcal{E}$ and for all $ag_j \in Q$, $(ag_i, Q, g) \in out^*(ag_j)$.

It can be seen that both the coalitions in Figure 1 (a) and (b) satisfy Definition 2.

However, the do-ut-des property does not consider the possibility that a coalition can be decomposed in smaller sub-coalitions which can be formed independently. Forming smaller coalitions can be preferred by agents because, e.g., involving less agents they reduce the risk of defections, are easier to monitor, less expensive to form by means of agreements, less trust is required among all the agents, etc. The coalitions in Figure 1 (a) and (b) show two cases of this fact. Consider in Figure 1 (a) the sub-coalitions C_1 , involving only agents ag_1 and ag_2 and the relative arcs, and C_2 , involving only agents ag_3 and ag_4 and the relative arcs. As the agents involved in C_1 are not interested in the goals achieved in C_2 and vice versa, the two sub-

coalitions can be formed independently.

Concerning Figure 1 (b), C denotes the whole coalition, C_1 the coalition consisting of the nodes ag_1 and ag_2 and the two arcs labelled with the goals g_1 and g_2 . C_2 denotes the coalition consisting of the nodes ag_3 and ag_4 and the remaining arcs labelled with the goals g_3 and g_4 . The difference is that in Figure 1 (a) both ag_1 and ag_2 are indifferent between C_1 and the whole coalition, while in Figure 1 (b) ag_2 is not indifferent between C_1 and C since in C it receives the goal g_4 which is not provided in C_1 .

However, ag_1 is indifferent between C_1 and C and ag_3 is indifferent between C_2 and C as they receive and have to obtain the same goals in both coalitions. Thus, if they agree to C , then they would agree respectively to C_1 and C_2 . When agent ag_2 wants to propose to ag_1 and ag_3 to form coalitions, it has to decide whether to propose C to both agents or C_1 and C_2 separately. It knows that if they agree to C , then they would also agree respectively to C_1 and C_2 . Agent ag_2 chooses C_1 or C_2 since forming one of them does not affect the possibility for ag_2 to reach an agreement on the other sub-coalition, and C_1 and C_2 are individually more reliable to succeed with respect to the whole C - as they involve individually less agents.

The i-dud property consists of three conditions. Given a coalition C , the first condition is the condition (1) of the do-ut-des property. The second condition strengthens the condition (2) of do-ut-des property by imposing that for each agent ag_i in C , $out^*(ag_i) = \mathcal{E}$. Thus, a coalition cannot be decomposed in two subgraphs which are disconnected as in Figure 1 (a). Finally, the third condition takes into account the case shown in Figure 1 (b): there does not exist an agent ag_i and a bi-partition $\mathcal{O}_1, \mathcal{O}_2$ of $out(ag_i)$ - where we assume that bi-partitions are composed by nonempty sets - such that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$. The idea underlying this third condition is that if $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$, then no agent $ag_j \neq ag_i$ involved in \mathcal{O}_1^* would be interested in one of the goals achieved in \mathcal{O}_2^* and vice versa. So, ag_i can deal separately with the formation of these two sub-coalitions.

Definition 3 A coalition $\langle \mathcal{V}, \mathcal{E} \rangle$ satisfies the i-dud property iff for all the agents $ag_i \in \mathcal{V}$, (cond1) there do not exist two commitments $(Q, g), (Q', g) \in C$ such that $Q \neq Q'$, (cond2) $out^*(ag_i) = \mathcal{E}$ and (cond3) there does not exist a bi-partition $\mathcal{O}_1, \mathcal{O}_2$ of $out(ag_i)$ such that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$.

Considering again the coalitions in Figure 1 (a) and (b), as expected, they both do not satisfy the i-dud property. In Figure 1 (a), the first condition of Definition 3 is not satisfied as, for example, $out^*(ag_1) = \{(ag_1, \{ag_2\}, g_1), (ag_2, \{ag_1\}, g_2)\} \subset \mathcal{E}$. In Figure 1 (b), the third condition of Definition 3 is not satisfied. Indeed, considering the bi-partition of ag_2 composed by $\mathcal{O}_1 = \{(ag_2, \{ag_1\}, g_2)\}$ and $\mathcal{O}_2 = \{(ag_2, \{ag_3\}, g_4)\}$, we have that $\mathcal{O}_1^* = \{(ag_2, \{ag_1\}, g_2), (ag_1, \{ag_2\}, g_1)\}$ and $\mathcal{O}_2^* = \{(ag_2, \{ag_3\}, g_4), (ag_3, \{ag_2\}, g_3)\}$. Thus, $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$.

A labeled AND-graph can represent a potential coalition consisting of all, or a large part of, the opportunities of collaboration in a multiagent system. So we would like to establish not only if the whole coalition is admissible or not, but also which of its sub-coalitions are admissible to be formed. Figure 2 (a) shows a quite complex coalition that does not satisfy the i-dud property. Indeed, given the bi-partition of $out(ag_1)$ $\mathcal{O}_1 = \{(ag_1, \{ag_4, ag_5\}, g_1)\}$ and $\mathcal{O}_2 = \{(ag_1, \{ag_3\}, g_6)\}$, it can be verified that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$. Figure 2 (b), (c), (d) and (e) show all the sub-coalitions of Figure 2 (a) which satisfy the i-dud property. These coalitions clearly satisfy also the do-ut-des property. However, since the do-ut-des property seeks only the reciprocity in a coalition, any composition of coalition (e) with one of the coalitions (b), (c) and (d) satisfies the do-ut-des

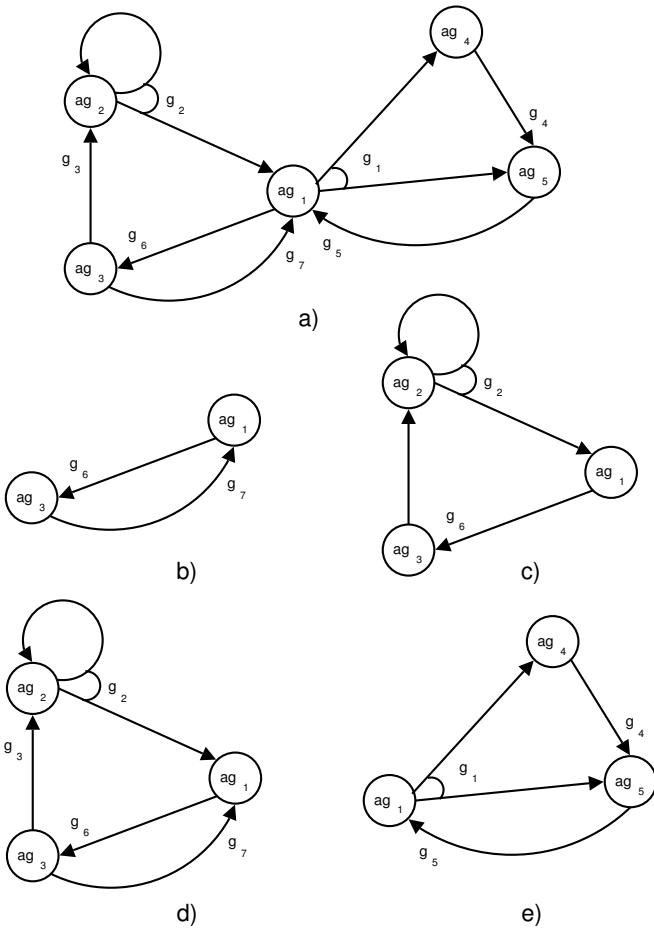


Figure 2. A complex coalition (a) and the sub-coalitions which satisfy the i-dud property (b), (c), (d) and (e).

property as well.

Given a coalition C , Definition 3 could be used *tout court* in order to design an algorithm which find all the sub-coalitions satisfying the i-dud property (C included). However, Definition 3 requires to verify, for each agent ag_i involved in C , a condition on the set of bi-partitions of $out(ag_i)$. The number of bi-partitions of a set A is equal to the Stirling number $S(n, 2) = 2^{n-1} - 1$, where n is the cardinality of A . Therefore, the problem to verify if just C satisfies the i-dud property would increase in complexity exponentially with the cardinality of $out(ag_i)$. For this reason we consider an alternative approach in order to make at least the verification of a single coalition tractable.

We reformulate cond2 as a property of strong connectivity of a directed graph. We define a direct graph $\mathbf{G}[C] = \langle \mathbf{V}, \mathbf{E} \rangle$ relative to the coalition $C = \langle \mathcal{V}, \mathcal{E} \rangle$ as follows: the set of nodes \mathbf{V} is equal to the set \mathcal{V} of agents involved in C and $(ag_i, ag_j) \in \mathbf{E}$ if and only if there exist a goal g and a group of agents Q such that $(ag_i, Q, g) \in \mathcal{E}$ and $ag_j \in Q$. It is easy to see that if $(ag_j, Q, g) \in out^*(ag_i)$, then there exists a path in $\mathbf{G}[C]$ from ag_i to ag_j . So, since each agent is involved in the achievement of at least a goal, the condition that $out^*(ag_i) = \mathcal{E}$ is equivalent to say that $\mathbf{G}[C]$ is strongly connected, i.e. for each pair of nodes ag_i and ag_j there exists a path from ag_i to ag_j . Given a generic directed graph \mathbf{G} , we call the strongly con-

nected components of \mathbf{G} the maximal strongly connected sub-graphs which contains at least one arc.

Now we consider how to reformulate cond3. Under the assumption that $\mathbf{G}[C]$ is strongly connected, cond3 is closely related to the notion of biconnectivity for undirected graphs. An undirected graph G is biconnected if and only if it is connected and for all triples of distinct nodes ag_i, ag_j and ag_k , there exists a path p connecting ag_j and ag_k such that ag_i is not in p . In the contrary case, ag_i is called an articulation node [9]. As for strongly connected components of a directed graph, the biconnected components of an undirected graph G are the maximal biconnected subgraphs of G which contain at least one arc. It is easy to see that two distinct biconnected components share at most one node and, if so, this node is an articulation node.

Starting from the directed graph $\mathbf{G}[C] = \langle \mathbf{V}, \mathbf{E} \rangle$, we define an undirected graph $G[C] = \langle V, E \rangle$ as follows: $V = \mathbf{V}$ and, for $ag_i \neq ag_j$, $\{ag_i, ag_j\} \in E$ if and only if (ag_i, ag_j) or (ag_j, ag_i) are in \mathbf{E} . The following theorem shows that the fact that cond3 is not satisfied is indicated by the presence of an articulation node ag_i .

Theorem 1 Let $C = \langle \mathcal{V}, \mathcal{E} \rangle$ be coalition such that $\mathbf{G}[C]$ is strongly connected, if there exists an agent $ag_i \in \mathcal{V}$ and a bipartition $\mathcal{O}_1, \mathcal{O}_2$ of $out(ag_i)$ such that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$, then ag_i is an articulation node of $G[C]$.

proof: Assume that there exists an agent $ag_i \in \mathcal{V}$ and a bipartition $\mathcal{O}_1, \mathcal{O}_2$ of $out(ag_i)$ such that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$ and, per absurdum, ag_i is not an articulation node.

Strong connectivity of $\mathbf{G}[C]$ assures that (ass1) there exist two agents, say ag_1 and ag_2 , such that ag_i, ag_1 and ag_2 are distinct and ag_1 is involved in \mathcal{O}_1^* and ag_2 is involved in \mathcal{O}_2^* , and (ass2) $\mathcal{O}_1^* \cup \mathcal{O}_2^* = \mathcal{E}$. Since ag_i is not an articulation node, there exists an undirected path p connecting ag_1 and ag_2 such that ag_i is not a node of p . For (ass2) and condition (1) of Definition 1, each node in the path is an agent in \mathcal{O}_1^* or \mathcal{O}_2^* . Thus, starting from ag_1 it is possible to walk through p until an agent ag_h is in \mathcal{O}_1^* and the successor ag_k is in \mathcal{O}_2^* . The presence of an undirected arc connecting ag_h to ag_k means that one of them is involved in set out of the other one. Without loss of generality we assume that there exists a set of agents Q and a goal g such that $ag_h \in Q$ and $(ag_k, Q, g) \in \mathcal{E}$. This means that $out(ag_h)$ is contained in both \mathcal{O}_1^* and \mathcal{O}_2^* . For strong connectivity of $\mathbf{G}[C]$, we also have that $out^*(ag_h) = \mathcal{E}$ and hence $out(ag_h)$ is not empty, then $\mathcal{O}_1^* \cap \mathcal{O}_2^* \neq \emptyset$ against the hypothesis. \square

So, if $\mathbf{G}[C]$ is biconnected (i.e. it does not have articulation points), then it satisfies cond3. However, the inverse implication of Theorem 1 does not hold and cond3 can be satisfied even if $\mathbf{G}[C]$ has an articulation point ag_i . This is due to the fact that the undirected graph $G[C]$ breaks an AND-arc in several undirected arcs, so the biconnected components sharing ag_i may not correspond to any bipartition of $out(ag_i)$. Figure 3 considers this fact. Figure 3 (b) represents the undirected graph $G[C]$ of the coalition C in Figure 3 (a). There exist two biconnected components of $G[C]$, one for each arc, sharing ag_1 as articulation node. However, both arcs $\{ag_1, ag_2\}$ and $\{ag_1, ag_3\}$ correspond to the AND-arc $(ag_1, \{ag_2, ag_3\}, g_1)$, thus they have to be considered as a single component because $\{(ag_1, \{ag_2, ag_3\}, g_1)\}^*$ contains both of them. Being $out(ag_1)$ equal to $\{(ag_1, \{ag_2, ag_3\}, g_1)\}$, there does not exist a bi-partition $\mathcal{O}_1, \mathcal{O}_2$ of $out(ag_1)$ such that $\mathcal{O}_1^* \cap \mathcal{O}_2^* = \emptyset$. Thus, C satisfies cond3.

This property holds in general, when some biconnected components of $G[C]$ contain some arcs which correspond to the same AND-arc of C , they are considered as a single component. If this grouping process ends with a single component consisting of the whole undirected graph $G[C]$, then no bi-partitions of $out(ag_1)$ falsify cond3.

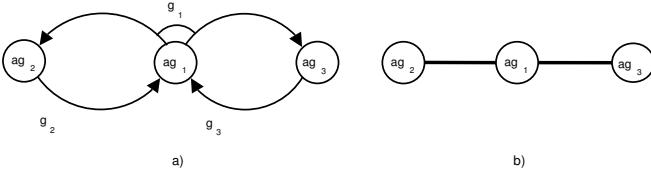


Figure 3. A labeled AND-graph and the corresponding undirected graph.

Algorithm 1: FIND-I-DUD

Data: $C = \langle \mathcal{V}, \mathcal{E} \rangle$.
Result: I_DUD , the set of sub-coalitions of C that satisfy the i-dud property

```

1  $I\_DUD \leftarrow \emptyset;$ 
2  $NDC = \text{NO-DUPL-COMMITMENTS}(C);$ 
3 forall  $C' \in NDC$  do
4    $I\_DUD \leftarrow I\_DUD \cup \text{FIND-2-3}(C');$ 
```

3 The algorithm for finding coalitions

In this section we design a procedure FIND-I-DUD (see Algorithm 1) which finds all the sub-coalitions of a coalition C satisfying the i-dud property (C included). We use the reformulation of cond2 and cond3 in Definition 3 in terms of strong connectivity of directed graphs and biconnectivity of undirected graphs. By doing so, we also decompose our problem as much as possible in well known problems in graph theory.

The variable I_DUD in line 1 stores the set of sub-coalitions of C which satisfy the i-dud property. In line 2 NO-DUPL-COMMITMENTS checks in \mathcal{E} the presence of commitments with the same goal but assigned to different sets of agents (cond1) and it returns the set NDC of all combinations C' obtained from C by deleting all the duplicated commitments except one. This way, the sub-coalitions in NDC are the maximal sub-coalitions which satisfy cond1. Since all their sub-coalitions satisfy this condition as well, we do not need to check recursively this condition on them, i.e. NO-DUPL-COMMITMENTS can be run only once.

For each coalition in NDC the procedure FIND-2-3 is run (lines 3-4). FIND-2-3 (Algorithm 2) takes as input a coalition C - which satisfies cond1 - and it returns the set of sub-coalitions of C , C included, that satisfy cond2 and cond3. As we already have checked cond1, we can add the results of FIND-2-3 to the set I_DUD .

The variable S stores the subsets of C that satisfy cond2 and cond3, in line 1 this variable is initialized to the empty set.

In line 2 SC-COMPONENTS calculates the strongly connected components SCC of $G[C]$ - algorithms for this procedure are well known [5, 9]. Three cases are distinguished.

Case 1: $G[C]$ has no strongly connected components. Since strong connectivity is a necessary condition for the satisfaction of cond2, no sub-coalitions of C satisfy the i-dud property. Therefore, S is empty.

Case 2: $G[C]$ is not strongly connected, but there exist some strongly connected components. In this case only the sub-coalitions of C such that the relative directed graphs are subgraphs of a strongly connected component can satisfy cond2. Therefore, in lines 7-10, for each strongly connected component, the maximal labeled AND-graph $\langle \mathcal{V}', \mathcal{E}' \rangle$, included in the component, is constructed. The func-

Algorithm 2: FIND-2-3

Data: $C = \langle \mathcal{V}, \mathcal{E} \rangle$.
Result: S , the set of sub-coalitions of C that satisfy cond2 and cond3.

```

1  $S \leftarrow \emptyset;$ 
2  $SCC \leftarrow \text{SC-COMPONENTS}(G[C]);$ 
3 switch do
4   case  $G[C]$  has no strongly connected components
5      $S \leftarrow \emptyset;$ 
6   case  $G[C]$  is not strongly connected, but it has some
7     strongly connected components
8     forall  $\langle \mathcal{V}, \mathcal{E} \rangle \in SCC$  do
9        $\mathcal{V}' \leftarrow \mathcal{V};$ 
10       $\mathcal{E}' \leftarrow \{(ag_i, Q, g) \in \mathcal{E} \mid ag_i \in \mathcal{V} \wedge Q \subseteq \mathcal{V}\};$ 
11       $S \leftarrow S \cup \text{FIND-2-3}(\langle \mathcal{V}', \mathcal{E}' \rangle);$ 
12   case  $G[C]$  is strongly connected
13      $(BC, A\_NODES) \leftarrow \text{BC-COMPONENTS}(G[C]);$ 
14     forall  $(ag_i, Q, g) \in \mathcal{E}$  s.t.  $ag_i \in A\_NODES$  do
15        $BC' \leftarrow \{\langle \mathcal{V}', \mathcal{E}' \rangle \in BC \mid \{ag_i, ag_j\} \in \mathcal{E}' \text{ with } ag_j \in Q\};$ 
16        $BC \leftarrow [BC \setminus BC'] \cup \{\bigcup BC'\};$ 
17     if  $|BC| = 1$  then
18        $S \leftarrow \{C\};$ 
19       forall  $(Q, g) \in C$  do
20          $C' \leftarrow C \setminus \{(Q, g)\};$ 
21          $S \leftarrow S \cup \text{FIND-2-3}(C');$ 
22     else
23       forall  $\langle \mathcal{V}, \mathcal{E} \rangle \in BC$  do
24          $\mathcal{E}' \leftarrow \{(ag_i, Q, g) \in \mathcal{E} \mid ag_i \in \mathcal{V} \wedge Q \subseteq \mathcal{V}\};$ 
25          $\mathcal{V}' \leftarrow \mathcal{V};$ 
26          $S \leftarrow S \cup \text{FIND-2-3}(\langle \mathcal{V}', \mathcal{E}' \rangle);$ 
27 return  $S;$ 
```

tion FIND-2-3 is recursively called on $\langle \mathcal{V}', \mathcal{E}' \rangle$ and its output is added to S .

Case 3: $G[C]$ is strongly connected, therefore, cond2 is satisfied. It remains to check cond3 and for complexity reasons we use the characterization by means of the biconnected components of $G[C]$ (see Section 2).

In line 12 the set of biconnected components BC and the set of articulation points A_NODES are calculated. In lines 13-15 FIND-2-3 checks, for each articulation node ag_i , if there exists an AND-arc (ag_i, Q, g) such that the other agents in Q are involved in two, or more, biconnected components, then these biconnected components replaced with their union (see Figure 3). In the case we end with a single component, $|BC| = 1$, C satisfies also cond3 and it is added to S . Then, in lines 18-20, the sub-coalitions C' obtained removing a single commitment (Q, g) from C are constructed and FIND-2-3 is recursively called on them. If $|BC| > 1$, then C does not satisfy cond3. Also all the subsets C' of C such that $G[C']$ is not included in a component of BC cannot satisfy cond3, therefore for each component $\langle \mathcal{V}, \mathcal{E} \rangle$ in BC , the maximal subgraph of C included in $\langle \mathcal{V}, \mathcal{E} \rangle$ is selected. FIND-2-3 is recursively called on C' and the output is added to S (lines 22-25). Finally, S is returned, line 27.

4 Complexity of the algorithm

In this section we discuss the complexity of FIND-I-DUD. First of all, we show that the problem of checking if a coalition satisfies the i-dud property is tractable. Algorithms FIND-I-DUD and FIND-2-3 can be easily modified to simply check if a given coalition satisfies the i-dud property. First, in FIND-I-DUD we replace the FOR statement with an IF-THEN-ELSE statement which returns false if C does not satisfy cond1, it calls FIND-2-3 on C , otherwise. In FIND-2-3 we replace lines 5,7-10, 22-25 with an instruction returning false, and lines 17-20 with an instruction returning true. We denote with m the number of agents involved in C and with l the number of arcs in $G[C]$. The procedure SC-COMPONENTS takes a time proportional to l [5]. In the case $G[C]$ is not strongly connected, C does not satisfy the i-dud property and FIND-2-3 returns false. In the contrary case, the procedure BC-COMPONENTS is called on the undirected graph $G[C]$.

Also BC-COMPONENTS can be executed in a time that is proportional to $|E|$ and, since $|E| \leq l$, so far Algorithm 2 has a complexity that is proportional to l . We have to consider now the complexity of the cycle corresponding to the lines 13-15. The number of iterations of the cycle 13-15 is less than l . The instruction in line 14 has as upper bound m , assuming that, during the execution of BC-COMPONENTS, a data structure is stored associating each arc with the biconnected component in which it is included. Since the sets of arcs of two distinct biconnected components are disjoint, also the instruction in line 15 can be performed in time proportional to the set of distinct biconnected components found in line 12, which has an upper bound in m . Therefore, the cycle 13-15 has an upper bound in $O(l \cdot m)$. Since $O(l \cdot m)$ is an upper bound also to check cond1, it is an upper bound for the problem to verify if a coalition satisfies the i-dud property.

With respect to the original problem to find all the sub-coalitions of C that satisfy the i-dud property, consider that C satisfies cond1 and it contains only AND-arcs as $(ag_i, \{ag_j\}, g)$. In this case we can represent C as a directed graph $G[C]$, where each arc (u, v) univocally corresponds to a goal. We show that a single run of FIND-2-3 is not computationally tractable. FIND-2-3 finds all the sub-coalitions of a coalition $G[C]$ that satisfy cond2 and cond3. This requires to find in particular all the subgraphs of $G[C]$ that are the strongly connected subgraphs and such that the relative undirected graph, $G[C]$, is biconnected. Since an hamiltonian cycle in $G[C]$ - if any exists - satisfies the previous two conditions, FIND-2-3 has to find a set of subgraphs which contains all the hamiltonian cycles of $G[C]$. Thus, this problem is exponential with respect to the number of arcs l . In contrast, since checking if a subgraph of $G[C]$ is an hamiltonian cycle is linear with the cardinality of the nodes V , also the problem of finding an hamiltonian cycle would be polynomial with respect to number of arcs.

In the case a coalition C can be represented by the corresponding directed graph $G[C]$, the set of subgraphs to check is equal to 2^l . However, if C does not satisfy the i-dud property, then, either $G[C]$ is not strongly connected or $G[C]$ has more than one component as calculated in the lines 12-15. In both cases FIND-2-3 is called directly on the subgraphs calculated respectively in lines 8-9 and 23-24. So, if there are k of these subgraphs, each of them with l_i arcs, we have that the number of the graph which remain to be verified is $2^{l_1} + \dots + 2^{l_k}$ instead of (approximately from below) $2^{l_1+\dots+l_k} - 1$. In the worst case $G[C]$ is not strongly connected and it has one strongly component with $l-1$ arcs. In this case 2^{l-1} graphs remain to be verified instead of $2^l - 1$.

We note that this fact occurs not only once, but every time a sub-coalition of C does not satisfy the i-dud property. Moreover, if C is a proper AND-graph this phenomenon can be amplified by the fact that when an AND-arc is removed in line 8, it may disconnect a strongly connected component of $G[C]$.

Returning to the coalition C in Figure 2 (a), the number of AND-arcs is 7, so *a priori* $2^7 = 128$ sub-coalitions should be checked by the algorithm FIND-I-DUD. However, C does not satisfy cond3 and, after BC results to be greater than 1 in line 16 of FIND-2-3, FIND-2-3 is called on the sub-coalitions in Figure 2 (d) and (e). The first sub-coalition has 4 AND-arcs and the second one has just 3 arcs. So after a single call of FIND-2-3, it remains $2^4 + 2^3 = 24$ sub-coalitions instead of 127. It can be shown that the total number of sub-coalitions checked is equal to 16, i.e. only the 12,5% of the number of all sub-coalitions of C .

5 Conclusion

In this work we define a criterion of admissibility for coalition formation which is based on the representation of a coalition as a net of exchanges [4]: the i-dud property. This property refines the do-ut-des property [2] by taking into account the fact that two distinct coalitions cannot be considered a whole coalition if they can be formed independently. This condition arises from the fact that agents prefer to form small coalitions because, as coalitions spring from unanimously agreements, the more are the agents involved in a coalition the more is the risk that one of them gives up joining it.

The i-dud property inherits from the do-ut-des property the fact that it uses only the internal topology of exchanges to check the admissibility of a coalition. Approaches based on Cooperative Game Theory, as [7, 10], abstract from this internal structure, and hence they need to compare a coalition with the other possible coalitions in order to establish its admissibility. This way, also the problem to see if a coalition is admissible, applying for example the notion of core, is intractable.

REFERENCES

- [1] R. Alur, T.A. Henzinger, and O. Kupferman, ‘Alternating-time temporal logic’, *Journal of ACM*, **49**(5), 672–713, (2002).
- [2] G. Boella, L. Sauro, and L. van der Torre, ‘Admissible agreements among goal-directed agents’, in *Proc. of 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT’05)*, Paris, France, (2005).
- [3] C. Castelfranchi, ‘Founding agents’ “autonomy” on dependence theory’, in *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI’00)*, pp. 353–357, Berlin, Germany, (2000).
- [4] R. Conte and J. S. Sichman, ‘Dependence graphs: Dependence within and between groups’, *Computational & Mathematical Organization Theory*, **8**(2), 87–112, (2002).
- [5] T. Cormen and C. Leiserson R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
- [6] S. Kraus and T. Plotkin, ‘Algorithms of distributed task allocation for cooperative agents’, *Theoretical Computer Science*, **242**(1-2), 1–27, (2000).
- [7] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohm, ‘Coalition structure generation with worst case guarantees’, *Artificial Intelligence*, **111**(1-2), 209–238, (1999).
- [8] L. Sauro, *Formalizing Admissibility Criteria in Coalition Formation Among Goal-directed Agents*, Ph.D. dissertation, 2006.
- [9] R. Tarjan, ‘Depth-first search and linear graph algorithms’, *SIAM Journal of Computation*, **1**(2), 146–160, (1972).
- [10] M. Wooldridge and P.E. Dunne, ‘On the computational complexity of qualitative coalitional games’, *Artificial Intelligence*, **158**(1), 27–73, (2004).

Advanced Policy Explanations on the Web¹

P. A. Bonatti and D. Olmedilla and J. Peer²

Abstract. The frameworks for protecting security and privacy can be effective only if common users—with no training in computer science or logic—increase their awareness and control over the policy applied by the systems they interact with. Towards this end, we introduce a mechanism for answering *why*, *why-not*, *how-to*, and *what-if* queries on rule-based policies for trust negotiation. Our framework is *lightweight* and *scalable* but it fulfills the main goals of modern explanation facilities. We adopt a novel *tabled explanation structure*, that simultaneously shows local and global (intra-proof and inter-proof) information, thereby facilitating navigation. Answers are focussed by removing irrelevant parts with suitable heuristics.

1 Introduction

The area of trust management—and in particular trust negotiation (TN)—is intersecting semantic web issues. In recent approaches [2, 5], software agents communicate their security and privacy requirements by exchanging policies formulated as rules, that is, simple ontologies. In this case, semantic descriptions concern the semantics of access control and information disclosure, which constitutes part of service and user agent semantics.

There is increasing awareness that advanced security and privacy techniques cannot be effective unless users are able to understand and possibly personalize the policy enforced by the systems they interact with (cf. [3] and the CUPS project on <http://cups.cs.cmu.edu/>). In order to enhance user awareness and control on policies, researchers are advocating a form of *cooperative policy enforcement* where policy decisions can be inspected by non-specialized users, and negative responses are enriched with suggestions and explanations.

In this paper, we describe an *advanced explanation mechanism* designed to help users understand what rule-based policies prescribe and control. Our first contribution consists in a requirements analysis for explanations in the context of trust negotiation. Moreover, we define explanation mechanisms for *why*, *why-not*, *how-to*, and *what-if* queries. There are several novel aspects in our approach:

- We adopt a *tabled explanation structure* as opposed to more traditional approaches based on single proof trees. The tabled approach makes it possible to describe infinite failures (which is essential for *why not* queries).
- Our explanations show the outcome of different possible proof attempts and let users see both local and global proof details at the same time. Such combination of intra-proof and inter-proof information is expected to facilitate navigation across the explanation structures.

- We introduce suitable heuristics for focussing explanations by removing irrelevant parts of the proof attempts. Any-way, we provide a second level of explanations where all the missing details can be recovered, if desired.
- Our heuristics are *generic*, i.e. domain independent. This means that they require no manual configuration.
- The combination of tabling techniques and heuristics yields a completely novel method for explaining failure. In the past, the problem has been ignored or formulated differently (e.g., by suggesting how to complete proofs by introducing new facts [4]).

Moreover, we aim at a *lightweight* and *scalable* explanation mechanism, that fits the requirements of web applications.

This paper is structured as follows. First, in Section 2, we recall the latest developments on explanations for expert systems. Then we outline in Section 3 the requirements for explanations in the context of trust negotiation, and briefly compare the two frameworks. Section 4 briefly introduces trust negotiation in our context and anticipates the functionalities of the explanation modules by means of a reference scenario. Then the explanation mechanisms are formally defined in two steps: First, the internal structures are defined (Section 5), then we describe how to render the explanation in natural language (Section 6). Section 7 concludes the paper with a final discussion of the results.

2 State of the Art

Integrated Explanation Facilities (IEFs) have been a goal for the development of intelligent systems early on. During the last fifteen years, a variety of design approaches for explanation facilities have been proposed, often classified as *second generation* frameworks by the literature [14, 6]. Examples of second generation IEFs include EES, the Explainable Expert System [11], the Mission Planning Assistant (MPA) [13, 12], the Reconstructive Explainer Rex [14]. These frameworks manage to decouple reasoning and explanation, whith the purpose of creating high quality explanations, at the cost of producing and synchronizing two versions of knowledge, one for reasoning and one for explaining. This is one of the main reasons that forced us to depart from these approaches (see next section).

From the Semantic Web perspective, the most comprehensive work on this new frontier is Inference Web (IW) [9, 10], a toolkit that aims at providing generic explanation tools for (Semantic) Web based systems. We could not use IW's facilities for our purposes because there is no support for explaining infinitely failed derivations. To navigate why-not queries we found it useful to explore multiple proof attempts simultaneously, while IW APIs are designed to manipulate single proofs. Moreover, IW's approach at removing the parts of the proof irrelevant to explanations (based on derived inference rules)

¹ Partially supported by REWERSE, IST-2004-506779.

² Universities of Naples, Hannover, and St. Gallen, respectively. D. Olmedilla is also a member of L3S. Contact: bonatti@na.infn.it

is not suitable for the kind of ellipsis and focussing we need, that we had to achieve by introducing a notion of *cluster* (see Section 5) and by exploiting predicate dependency graphs.

Finally, in the context of security, the KNOW system [7] focusses on the provision of feedback after a request is denied. However, KNOW does not return an explanation but a set of changes to the policy that would make it fulfilled. Similarly, the WhyNot system [4] suggests which sets of facts might be added to the system to make a given goal succeed, based on some heuristic weights calculated on proofs.

3 Requirements

Since TN frameworks can be applied in many application domains, rule-based policies almost always refer to domain specific concepts and application specific information sources. So a TN framework has to be suitably instantiated for each application by defining the set of application specific predicates, and interfacing them with legacy software and data. *Almost no further effort should be added to the framework instantiation phase.*

A second requirement is related to scalability: *explanations should not increase significantly the computational load of the servers* which is already increased by rule manipulation.

The first requirement is incompatible with the methodology of [1], that prescribes an actor and five distinct phases entirely devoted to the development of an independent explanation module equipped with an ad-hoc domain ontology and special rules for creating explanation-related data structures.

The second requirement is incompatible with the methodology of [1], too. The special rules for creating explanation-related data structures are meant to be executed during the reasoning process and may potentially affect performance.

A third requirement, instead, is shared with second generation explanation systems: explanations should be closer to the users' problem solving strategies than the system's automated reasoning strategy. The reason is that policy explanation systems should be understood by any user.

According to modern explanation approaches, the support for explanation presentation is characterized by the following features [9]:

- **Methods for asking for explanations.** We have identified the following kinds of queries: *why/why not, how to, what if*, that may be asked before, during, and after a negotiation to understand which pieces of information are actually used (some information may be unnecessarily released), what remains to be done. The same queries can be used to inspect and monitor policies.
- **Methods for breaking up proofs into manageable pieces.** The local view (the rules that directly apply to a given goal), should be enriched with global information such as the different answer substitutions of each subgoal, to help users in deciding which proof and which proof branches should be visited next.
- **Methods for pruning proofs and explanations to highlight relevant information.** The negotiation protocol determines what is relevant: Peers fulfil conditions by submitting credentials and other information, so the pieces of information that have been submitted and those that have not determine the focus of attention. State predicates constitute another kind of crucial information in TN, be-

cause their semantics is often *blurred* [2], i.e. it is not communicated to the client (either for privacy reasons or for efficiency). This raises the need for special explanations.

- **Methods and user interfaces for proof and explanation navigation:** We see a proof as a (potentially cyclic) hypertext, whose links help the user in exploring single as well as alternative proofs and proof attempts.
- **Different presentation formats.** Natural language is an appealing, user-friendly format. It can be complemented by graphical representations.
- **Methods for obtaining justifications for conflicting answers.** Today the languages involved in TN are not expressive enough to derive contradictions, so this aspect is currently marginal but this may change in the future.

Finally, the entities referred by the policies should be denoted in a user-friendly way. Their internal encoding (XML, object handles, etc.) is generally not suitable because it is meaningless to most users.

4 TN and Explanations

We apply our techniques to PROTUNE [2], one of the most recent trust negotiation frameworks. In summary, each party makes decisions on access control and information disclosure according to a set of rules that entail decision atoms such as `allow(X)`. The rule language extends Datalog with syntactic sugar. Policy rules are extended with a time-dependent set of facts, including currently available credentials and declarations (sent by the other party), other negotiation-related information, user profiles, etc. The argument of `allow(X)` may refer to a service or it may denote a credential release, declaration release or the execution of some specified action. To explain its disclosure requirements to the other peers, each peer sends out the *rules* themselves, as a compact representation of all the possible ways of fulfilling the request. First, however, the rules are suitably filtered to protect the sensitive parts of the policy (the policy itself may be confidential). Explanations are built from filtered policies (so they can be built on the clients). The following scenario illustrates some of the explanations that our method produces from the filtered rules.

A digital library protects its resources according to the policy partially illustrated in Figure 1. John Smith tries to download a paper with file name "paper01234.pdf" and authenticates himself by providing an id credential. He receives the answer "permission denied" from the library service. To understand why, he sends a *why-not* query to the service. His personal assistant gathers the policies provided for why-not explanations and filters them highlighting the parts most relevant to the user, i.e. those requirements the user did not fulfill, and hiding some other aspects of the policy (e.g., those that do not depend on the user). The first output is:

I can't prove that it is allowed to download paper01234.pdf **because:**

- Rule [r₃] is not applicable:
there is no User such that
User is authenticated [details]
- and
rule [r₄] is not applicable:
there is no User such that
User is authenticated [details]

```

...
[r2] : allow(download(Resource)) ←
    public(Resource).
[r3] : allow(download(Resource)) ←
    authenticated(User),
    has.subscription(User, Subscription),
    available_for(Resource, Subscription).
[r4] : allow(download(Resource)) ←
    authenticated(User),
    paid(User, Resource).
...
[r6] : authenticated(User) ←
    id(Credential),
    Credential.name : User,
    Credential.public_key : K,
    challenge(K).
[r7] : authenticated(User) ←
    declaration([username = User, password = P]),
    passwd(User, P).
...

```

Figure 1. Digital Library Policy

moreover
there is no User such that
User has paid for paper01234.pdf [details]

Concise explanations (like the one presented above) do not show all the details and focus primarily on those conditions that depend on the user, giving him the opportunity to fulfill the conditions quickly. For example rule r_3 talks about subscriptions, but these details are omitted in the explanation above because if John is not authenticated it makes no sense to inspect his subscription. However, John could still request the full explanation by clicking on the [details] link.

John did disclose his id credential and wonders why the authentication failed. By clicking on that condition he eventually finds why (the definition of $\text{id}/1$ is not shown in Figure 1):

I can't find any Cred such that Cred is an id because:
c012 is a credential with
 type student-id and issuer Open University [details]
student-id is an id-type [details]
but
it is not the case that
Open University is trusted for id [details]

This explains why John's request has not been accepted: the certification authority (CA) 'Open University' on his credential is not among the trusted CAs for id credentials at the library service. Note that the above explanations anticipate the results of each subgoal (e.g. a failure, or the answer substitution fixing the credential's identifier, type and issuer). In case of multiple answer substitutions, the user can inspect the list of all answers of a subgoals and apply some, to focus on a subset of all proof attempts (see *refinement links* in the following sections).

John might ask for a complete explanation of the possible ways of obtaining the paper by issuing a *how-to* query. The result would be:

to make sure that it is allowed to download Resource
nothing needs to be done if
Resource is public [details]
alternatively
please make sure that for some User

User is authenticated	[details]
where, for some Subscription,	
User has subscription Subscription	
and	
Resource is available for Subscription	[details]
alternatively	
please make sure that for some User	
User is authenticated	[details]
and	
User has paid for Resource	[details]

Again, John can get a specific how-to explanation for each of the above conditions by clicking on it. The "make sure" part contains predicates that depend on user actions (including `credential`); they are identified via a standard *predicate dependency graph*. Due to space constraints, *what-if* queries, *why* queries, and "technical" (level 2) explanations that provide full proof details are not discussed in this paper.

5 Explanation Structures

To meet page limitations, we present here slightly simplified explanation structures that do not support blurred predicates (which involve notions similar to abductive explanations and verbalizations such as: "it might be the case that...").

We assume the reader to be familiar with the basics of logic programming, including the notions of *most general unifier* (mgu) and *(computed) answer substitution*. The reader is referred to [8] for these matters. There are some delicate technical aspects in handling substitutions, related to standardization apart. So we assume a function $\text{ans}_P^E(G)$ that for all programs P , all goals G , and all expressions E returns a complete (up to variable renaming) set of answer substitutions whose range does not contain any variable occurring in P, E, G .

Before formalizing explanations we tackle the problem of referring to structured objects, such as credentials, that are typically denoted by means of internal identifiers (*handles*) that have no meaning to the common user. We noted that the attribute atoms $obj.attr:val$ in a rule body are typically the relevant, characterizing attributes of the complex object whose handle is obj —see for instance the why-not explanation referring to credential c012 in Section 4. In a given negotiation, these attributes almost always identify a unique object when their values are fixed. So our idea is using these attributes as a key to identify the object. For this purpose, we look for subsets of the body called *clusters* that correspond to concepts with attributes (a sort of terminological expression) and treat them as a description of the complex object.

Definition 1 (Clusters) Let $L = p(t)$ where p is a unary predicate and t is a term. If $L \in \text{body}(r)$, then a cluster of L (in r) is a set containing L and some attribute subgoals $(u.a : v) \in \text{body}(r)$ with $u = t$. A cluster is complete if it contains all such subgoals. If t is a variable, then it is called the main variable of the cluster; L and p are called the main literal and the main predicate of the cluster, respectively.

Example 1 The policy rule instance r_6 in Figure 1 contains one complete cluster (the first three literals in the body) whose main literal is $\text{id}(\text{Credential})$. The cluster denotes "the id with name User and public key K". \square

Remark 1 Clusters are more flexible than key attributes because they can be applied also to those classes of objects that

have no key attributes. Moreover, clusters reduce the framework instantiation effort, because they need no manual intervention (while key attributes must be specified by knowledge engineers). Clusters are very useful in *why not* queries, where incomplete clusters allow to explain situations such as: “*there is an id with name J. Smith, but the public key is not k012*”.

While a single atom like `credential(X)` may have multiple solutions, often its cluster has just one answer (in this sense the attributes of X characterize X). By applying this substitution, we specify the credential we are talking about. The process of exhaustively applying substitutions when they are the unique answer of a cluster or a subgoal is formalized as a binary (rewrite) relation \rightarrow_U over annotated rules (r, θ) . The precise definition and the heuristics for the cases where different rewrite sequences yield different results³ are in the full report. The framework can be adapted to different heuristics simply by changing the definition of \rightarrow_U .

We are ready to formalize explanations. They are graphs (abstracting a hypertext) where each node illustrates: (i) the local context for a goal, that is, the rules whose head unifies with that goal; (ii) a global view of all the proof attempts, consisting in the answers of each rule body and subgoal thereof. Such answers provide a sort of *lookahead* on proof outcomes that may help the user in navigating the proof. We proceed by defining the explanation graph, starting with its nodes and entry points.

Definition 2 An explanation node for a program P is a finite set of pairs (r, θ) where $r \in P$ and θ is a substitution.

The explanation entry point for an atom A w.r.t. P , denoted by $\text{entry}_P(A)$, is the (unique) explanation node X for P such that

$$X = \{(r, \theta) \mid r \in P, A \text{ is unifiable with } \text{head}(r), \\ \text{and } (r, \text{mgu}(A, \text{head}(r))) \rightarrow_U (r, \theta)\}.$$

There are two kinds of navigation links: *detail links* and *refinement links*. Informally, the former expand the proof details by showing the rules that can be used to prove a subgoal; the latter apply answer substitutions locally to the rule to see the effects on the other subgoals and focus on a subset of all the possible proofs.

Now we can proceed with the formalization of navigation links. They lead directly to nodes where unique answers have been propagated, and consider only maximally general answer substitutions to reduce redundancy. So we need a function $\text{mg}(S)$ that for all sets of substitutions S returns the maximally general elements of S . In the following we extend \rightarrow_U to explanation nodes as follows:

$$X_1 \rightarrow_U X_2 \text{ iff } X_2 = \{(r, \theta') \mid \\ \text{for some } (r, \theta) \in X_1, (r, \theta) \rightarrow_U (r, \theta')\}.$$

Definition 3 (Level 1 navigation link) For all explanation nodes X_1 and X_2 for P we define:

detail links: $X_1 \xrightarrow{L} X_2$ iff for some $(r, \theta) \in X_1$ and some $L \in \text{body}(r)$, $\text{entry}_P(L\theta) \rightarrow_U X_2$;

refinement links: $X_1 \xrightarrow{\sigma} X_2$ iff for some $(r, \theta) \in X_1$ and some $L \in \text{body}(r)$, $\sigma \in \text{mg}(\text{ans}_P^r(L\theta))$ and $\{(r, \theta\sigma)\} \rightarrow_U X_2$

³ This may happen if $\text{body}(r)$ fails.

Example 2 Let P be the program illustrated in Figure 1 extended with time-dependent facts, and $A = \text{allow}(\text{'paper0123.pdf'})$. If the body literals of r_2 , r_3 , and r_4 have no solutions, then $\text{entry}_P(A) = \{(r_2, \theta), (r_3, \theta), (r_4, \theta)\}$ where $\theta = [\text{Resource} = \text{'paper0123.pdf'}$]. If only $\text{authenticated}(\text{User})$ has a unique solution, say, $[\text{User} = \text{'John'}]$, then $\text{entry}_P(A) = \{(r_2, \theta), (r_3, \theta'), (r_4, \theta')\}$ where $\theta' = [\text{Resource} = \text{'paper0123.pdf'}, \text{User} = \text{'John'}]$. Now suppose that the subgoal $\text{has.subscription}(\text{John}, \text{Subscription})$ of $r_3\theta'$ has two answer substitutions σ, σ' . The action of selecting and applying σ to r_3 is formalized by crossing the refinement link $\text{entry}_P(A) \xrightarrow{\sigma} \{(r_2, \theta), (r_3, \theta'\sigma), (r_4, \theta')\}$. Otherwise, the action of expanding the details of the subgoal $L = \text{authenticated}(\text{User})$ of $r_3\theta'$ is formalized by the detail link $\text{entry}_P(A) \xrightarrow{\sigma} \{(r_6, \gamma), (r_7, \gamma')\}$, where γ, γ' result from applying the unique answers of the subgoals of r_6 and r_7 . \square

Definition 4 A level 1 explanation graph for an atom A w.r.t. a partially specified program P is a minimal structure (V, E^D, E^R) closed under the following properties:

1. V contains $\text{entry}_P(A)$;
2. if for some $X_1 \in V$, $X_1 \xrightarrow{L} X_2$ then V contains exactly one variant \tilde{X}_2 of X_2 , and E^D contains an edge (X_1, L, \tilde{X}_2) ;
3. if for some $X_1 \in V$, $X_1 \xrightarrow{\sigma} X_2$ then V contains exactly one variant \tilde{X}_2 of X_2 , and E^R contains an edge $(X_1, \sigma, \tilde{X}_2)$.

Since our programs are basically Datalog programs, it can be shown that the explanation graph is always finite. It is cyclic in the presence of infinitely failed proofs, and the browser can highlight loops by marking the nodes that have already been visited. In the full paper, *explanation structures* are defined as explanation graphs labelled with suitably filtered answer substitutions (the *global*, possibly *inter-proof* information).

6 Verbalization

Explanation structures are translated into natural language sentences by instantiating text patterns. The verbalization patterns for application dependent literals must be supplied during the framework instantiation phase. They are typically formulated with simple PROTUNE metafacts like:

`is_authenticated(X).explanation : [X, is_authenticated]`

If necessary, proper rules (with nonempty body) can be used and it is also possible to define text patterns for negative literals. By default, variable names are taken from the original rule—which is the first element of the current node (r, θ) .

A cluster $C = \{L, t.a_1 : v_1, \dots, t.a_n : v_n\}$ of (r, θ) with main literal L is verbalized as a single condition:

`verb(L1\theta) with a1 v1, a2 v2, ..., and an vn.`

The rest of this section is implicitly formulated w.r.t. the following context:

- a node $X = \{(r_1, \theta_1), \dots, (r_z, \theta_z)\}$ of XG (the current node);
- an atom A (the one being currently explained), more general than $\text{head}(r_i\theta_i)$ for $1 \leq i \leq z$.

For all clusters or literals Y and for all substitutions θ , the verbalization of $Y\theta$ will be denoted by $\text{verb}(Y, \theta)$. For the sake of simplicity, the navigation links will not be shown.

We show the verbalization details for *why-not* queries (the details for the other queries are in the full paper). They start with:

I can't prove that $\text{verb}(A, \varepsilon)$ because: (if A is ground)

I can't find any <var list> such that $\text{verb}(A, \varepsilon)$ because: (if A is not ground).

The above *incipit* is followed by the verbalization of all $(r_i, \theta_i) \in X$. Each pair (r_i, θ_i) is formatted according to the following pattern:

Rule <rule identifier> is not applicable:
<Success Slot> but <Failure Slot>

where **but** is omitted if any of the two slots is missing. The Success Slot in its most complete version is verbalized as:

<quantification>
verb(Y_1, θ_i) and ...and verb(Y_m, θ_i) and verb(L_1, θ_i)
and ...and verb(L_n, θ_i)

where $Y_1 \dots Y_m$ are all the \subseteq -maximal clusters in $\text{body}(r_i \theta_i)$ with a single answer (i.e. $|\text{ans}_P(Y_i)| = 1$), and $L_1 \dots L_n$ are all the literals with a single answer (i.e. $|\text{ans}_P(L_j)| = 1$) that do not occur in $Y_1 \dots Y_m$.

If $\text{body}(r_i \theta_i)$ contains some failed clusters or literals (with zero answers), then the Failure Slot is verbalized as⁴

verb(not Z_1, θ_i) moreover...moreover
verb(not Z_j, θ_i) moreover verb(not L'_1, θ_i) more-
over...moreover verb(not L'_k, θ_i)

where $\{Z_1 \dots Z_j\}$, denoted by **minfailed**, consists of the \subseteq -minimal failed clusters in $\text{body}(r_i \theta_i)$, and $L'_1 \dots L'_k$ are the failed literals in $\text{body}(r_i \theta_i)$ that neither occur in **minfailed** nor belong to the set of attribute atoms $t.a:v$ such that t is the main variable of a singleton cluster in **minfailed**. With the latter condition, the attributes of non-existing objects are ignored.

Example 3 If no id has been provided, then (with this condition) the *why not* verbalization of r_6 is simply: “**there is no Credential such that Credential is an id**” as expected. A naive approach would have produced something like “**there is no Credential such that Credential is an id with name User and public.key K**”, which is misleading. □

If no clusters or literals fail, the Failure Slot is verbalized as

each of the following facts has some solution
<quantification> verb(L'_1) ... <quantification> verb(L'_k)
but there is no common solution

where $L'_1 \dots L'_k$ are the literals with two or more solutions in $\text{body}(r_i \theta_i)$. Via refinement links, the user may apply a chosen answer substitution and explore why it eventually fails (i.e. which other subgoals become failed after applying that substitution).

7 Conclusions

In TN it is possible to construct good explanations in response to *why*, *why not*, *how to*, and *what if* queries, using simple generic explanation strategies based on the intended meaning of a few core predicates with a special role in negotiations. The

only extra workload needed during the framework instantiation phase to support explanations consists in writing literal verbalization patterns. Moreover, the extra computational burden on the server can be limited to adding a few more rules to the filtered policies (the literal verbalization rules) because the explanations can be independently produced on the clients. Despite its simplicity, our explanation mechanism supports most of the advanced features of second generation explanation systems.

Another contribution of our work is a novel explanation structure combining local and global information about single and multiple proof attempts. The new explanation structures are analogous to the tables of tabled logic programming engines, so our work might be applied to execution tracing in tabled Prolog implementations like XSB. Our method provides an effective way of explaining infinitely failed proofs. Global proof information and heuristics help in focussing the explanation strictly on relevant details. The resulting explanations have no counterpart in previous literature.

REFERENCES

- [1] R. Barzilay, D. McCullough, O. Rambow, J. DeChristofaro, T. Koretsky, and B. Lavoie. A new approach to expert system explanations. In *9th International Workshop on Natural Language Generation*, pages 78–87. 1998.
- [2] P. A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *6th IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 14–23, jun 2005.
- [3] Bonatti et al. The reverse view on policies. In *Proc. of the ISWC Semantic Web Policy Workshop (SWPW)*, <http://ebiquity.umbc.edu/get/a/publication/215.pdf>, 2005.
- [4] H. Chalupsky and T. A. Russ. Whynot: debugging failed queries in large knowledge bases. In *14th national conference on Artificial intelligence*, pages 870–877, 2002.
- [5] R. Gavriloaie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *1st European Semantic Web Symposium*, volume 3053, pages 342–356, may 2004.
- [6] S. R. Haynes. *Explanation in Information Systems: A Design Rationale Approach*. PhD thesis, London School of Economics and Political Science, Dept. of Information Systems and Dept. of Social Psychology, 2001.
- [7] A. Kapadia, G. Sampemane, and R. H. Campbell. Know why your access was denied: regulating feedback for usable security. In *11th ACM conference on Computer and communications security*, pages 52–61, 2004.
- [8] J. Lloyd. *Foundations of logic programming*. Springer-Verlag, 1984.
- [9] D. L. McGuinness and P. P. da Silva. Explaining answers from the semantic web: The inference web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.
- [10] D. L. McGuinness and P. P. da Silva. Trusting answers from web applications. In *New Directions in Question Answering*, pages 275–286, 2004.
- [11] W. Swartout, C. Paris, and J. Moore. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert: Intelligent Systems and Their Applications*, 6(3):58–64, 1991.
- [12] M. C. Tanner, A. Keunecke, and B. Chandrasekaran. Explanation using task structure and domain functional models. In *Second Generation Expert Systems*, pages 586–613. 1993.
- [13] M. C. Tanner and A. M. Keuneke. Explanations in knowledge systems: The roles of the task structure and domain functional models. *IEEE Expert: Intelligent Systems and Their Applications*, 6(3):50–57, 1991.
- [14] M. R. Wick. Second generation expert system explanation. In *Second Generation Expert Systems*, pages 614–640. 1993.

⁴ In the following expression we identify $\text{not not } A$ and A .

Prevention of harmful behaviors within cognitive and autonomous agents

Caroline Chopinaud¹ and Amal El Fallah Seghrouchni² and Patrick Taillibert³

Abstract. Being able to ensure that a multiagent system will not generate undesirable behaviors is essential within the context of critical applications (embedded systems or real-time systems). The emergence of behaviors from the agents interaction can generate situations incompatible with the expected system execution. The standard methods to validate a multiagent system do not prevent the occurrence of undesirable behaviors during its execution in real condition. We propose a complementary approach of dynamic self-monitoring and self-regulation allowing the agents to control their own behavior. This paper goes on to present the automatic generation of self-controlled agents. We use the observer approach to verify that the agents behavior respects a set of laws throughout the system execution.

1 Introduction

The behavior of a multiagent system (MAS) emerges from the agents' interaction. Some emergent behaviors can lead the system to fail: these are **undesirable behaviors**. In our study, the autonomy is an essential specificity of cognitive agents. We consider the autonomy as the capacity of an agent to take its decision alone, without the help of another entity [1]. From the developer point of view, the agent autonomy requires to take into account the unpredictability of the agents' behavior. This view increases the possibilities of occurrence of undesirable behaviors.

The potential occurrence of undesirable behaviors could be critical for the MAS execution, especially in the context of embedded or real-time applications. To ensure the reliability of the multiagent systems, our work aims to provide a mechanism allowing the detection and the prevention of the occurrence of these undesirable behaviors, harmful to the system execution. A first approach could be the application to multiagent systems of widely used validation methods like: tests, model-checking and automatic demonstration. But, these methods cannot find all the errors in a system: the model-checking works on an abstraction of the system and its environment; the automatic demonstration is complex and heavy; tests cannot be exhaustive [3]. So, in order to detect the remaining errors, we propose to verify the system at runtime. From this perspective, the recent work of R. Paes & al. [12] proposes the use of laws to control the emergence of wrong behaviors in the context of open MAS. Their approach consists in monitoring the interaction of the agents thanks to an external entity and in describing the interaction protocols to detect violation. Similarly, the work of Klein and Dellarocas [7] proposes an approach of the exception handling thanks to a description of the

exceptions and external entities filtering the agents' communication. In the work of S. Hagg [6], sentinels are used to monitor the communication between the agents, build models of other agents and intervene according to given guidelines. To our knowledge, works about the surveillance of agents behaviors use external entities to monitor the agents and detect inconsistencies, particularly with respect to their communications and interactions. Moreover, most of the verification steps are done manually. The main advantage of our approach consists in the automatic creation of agents which are able to verify their behaviors from laws. This verification is based on the dynamic self-monitoring and self-regulation of each agent in order to prevent the system failure. We call this verification, the **agent control**.

The section 2 presents the agent control. The section 3 describes the laws and their use to control the agents by themselves. The section 4 details the whole process of the generation of a self-controlled agent. The section 5 describes the working of our approach and the section 6 introduces the multiagent case. Finally, the section 7 introduces a prototype implementing our approach while the section 8 concludes our paper.

2 Control of autonomous agents

The agent control is divided in three stages :

- The monitoring of the behavior of an agent or a group of agents.
- The detection of undesirable behaviors that could emerge from the execution of an agent or a group of agents.
- The regulation of the problematical agent(s).

2.1 Monitoring of an agent behavior

Monitoring the behavior of each agent in MAS in order to observe relevant events, allows to detect or prevent the occurrence of undesirable behaviors in the system [4]. Our work focuses on the observation of the MAS behavior at runtime. To observe the behavior of an agent, it is necessary to insert event detectors in the agent program. Although it is possible for a developer to insert manually the probes into the agent program, this kind of instrumentation is hard, time-consuming and prone to error [9]. We propose an automatic instrumentation of the programs of the agents, *i.e.* to insert a code of control to detect some events from the specification of *hooks* provided by the developers. The main interest of this automation is to simplify the work of the developer and to reduce the risk of errors during the instrumentation.

¹ LIP6, France, email: caroline.chopinaud@lip6.fr

² LIP6, France, email: amal.elfallah@lip6.fr

³ Thales Aerospace, France, email: patrick.taillibert@fr.thalesgroup.com

2.2 Detection of undesirable behaviors

Making an exhaustive model of all the MAS behaviors is difficult and costly because of its complexity (indeterminism, state explosion, distributed nature, interleaving between the autonomous agents behaviors). So we propose to abstract this model thanks to the definition of properties about the agents' behavior. These properties are expressed using norms [13], usually defined as constraints on the agents' behavior in order to guarantee a collective order. An autonomous agent decides to respect or not a norm depending on the actions it chooses to perform. We use **laws** to describe desired or dreaded behaviors or situations ; these laws represent significant or critical requirements on the system execution and could be specified by the customer. At the opposite of norms, laws are defined independently of the agents' design and are not taken into account by the agents during the decision process. In other words, the agents can act autonomously while our approach consists in verifying afterwards if the chosen action respects the laws. Consequently the agents' implementation and the laws/control description are distinguished. The detection of undesirable behaviors can be viewed as the detection of laws transgression.

2.3 Self-regulation of agents behaviors

When an occurrence of (a potential) undesirable behavior is (to be) detected, (*i.e.* when a law is transgressed), the agents concerned by the law must regulate their behavior by themselves thanks to their capacity of reasoning. The developer provides the agents with a set of strategies of regulation associated with the transgression of laws and, at runtime, the agents deduce the behavior to follow when they are informed of a law transgression. This behavior of regulation takes the transgression into account but also the main behavior of the agent, its current state, or even the other agents. Then, it seems quite complex to define these strategies independently of the agents' implementation, as for the laws. It is the reason why, to date, the strategies are defined directly in the agent program.

3 The control description

In our approach, the description of the control applied to a multiagent system comes down to the description of laws associated with the agents.

3.1 Level of description

For a sake of generality, we would like to provide a mechanism of control which is not specific to one agent model. Hence, the laws concern abstract concepts representing the application and the model(s) of agent(s) used. We propose an ontology composed of the following concepts: AGENT, FEATURES and ACTION. The FEATURES concept has the sub-concept of Object, Message, Goal, Plan and Knowledge. The ACTION concept has the sub-concepts of Creation_Agent, Sending_Message, Receipt_Message and Migration.

The designers of the system may freely extend this set of basic concepts to refine and to enrich the description of the model(s) of the agent(s) and the application. Moreover, in order to monitor the agent behavior, we have introduced previously that the developers can describe hooks (cf 2.1). These hooks define the relation between the concepts describing the agent model(s) and its (their) implementation (for more details see [2]).

3.2 A language of Law

To express the laws, we propose a language based on deontic operators, which are widely used in the context of norms. Our language applies to events and states about the agents, corresponding to the general basic concepts of FEATURES and ACTION, introduced in the previous part. An **event** can be the execution of an action or the change of a features value. A **state** represents the resulting state of an event. The expression of time or temporal relation between the events and states is crucial in our framework and it is taken into account in our language as we will see below. Our language distinguishes two kinds of law: prohibition and obligation. A prohibition law represents the unwanted states or events of an agent. It allows the detection of situations where an event or a state which must never occur, is going to happen. An obligation law represents the expected states or events of an agent. It allows the detection of situations where an expected event or state does not occur. In our approach a law is composed of three parts:

- **CONCERNED AGENT (CA):** the statement of agents concerned by the law, *i.e.* the agents subject to the law and agents used to describe the law application context.
- **DEONTIC ASSERTION (DA):** the description of what is obligatory or forbidden. It is a set of relationships between an agent and an event or a state.
- **APPLICATION CONDITIONS (APC):** the description of the law context. It is an expression describing when the DA must be respected relatively to a set of events or states.

The description of our language is as follows:

<i>LAW</i>	$::= (CA) (DA \langle APC \rangle)$
<i>CA</i>	$::= agent : AGENT \langle \text{and } PROP \rangle$
<i>DA</i>	$::= DEON_OP EXP$
<i>APC</i>	$::= TEMP_OP EXP \mid TEMP_OP EXP APC$
<i>DEON_OP</i>	$::= FORBIDDEN \mid OBLIGED$
<i>TEMP_OP</i>	$::= AFTER \mid BEFORE \mid IF$
<i>EXP</i>	$::= TERM \mid TERM AND EXP \mid TERM THEN EXP$
<i>TERM</i>	$::= EVENT \mid NOT EVENT \mid EVENT TIME$
<i>EVENT</i>	$::= agent do SMTM \langle \text{and } PROP \rangle \mid agent be SMTM \langle \text{and } PROP \rangle$
<i>TIME</i>	$::= -second \mid +second$

where SMTM represents a concept like ACTION or FEATURES; AGENT represents the concept of agent; PROP is a common attribute representing the properties on these concepts.

The laws are, actually, deontic formula expressed in dynamic deontic logic[10]. Knowingly, the language limits the possibilities of laws expressiveness to a sub-set of the dynamic deontic logic, allowing the description of behaviors which can be monitored. For example, a law can be expressed in this way: “(L1): *It is forbidden for an agent A1 to do an action ACT2 after an action ACT1 and before a time Sec*” and by using the language :

(L1): (A1 : Agent)
FORBIDDEN (A1 do ACT2) **AFTER** (A1 do ACT1) - Sec.

4 Self-controlled Agents

A main interest of our approach is the generation of agents being able to control their own behaviors. A self-controlled agent is obtained automatically from:

- the agent behavior program,
- the set of laws associated with the agent,
- the hooks between the concepts used in the laws and the agent model implementation.

A generated agent has a specific architecture of control allowing its monitoring and the detection of laws transgression thanks to the **observer mechanism** [5].

4.1 The observer

The observer mechanism is used in the context of real-time systems testing, on-line validation of parallel, or even distributed systems. The observer consists in a parallel execution of a program and a model of properties applied to the program execution. The model and the program are connected with control points. A controller checks on the model and the program execution are consistent. If the system execution does not match the model, the verification fails.

We propose to put this approach in place into the agents in order to provide them with the means to control their own behavior. A law is modeled by means of a Petri net [11] whose transitions are bound to the program with control points. The Petri nets representing each law are automatically generated (c.f. 4.3). We insert into the agent behavior program the control points and we generate a runnable agent with a specific architecture, using the observer approach. When the program execution meets a control point, the controller makes sure the tokens are in the right place at the right time in the corresponding net, and brings about some change in the model, accordingly.

4.2 The agent architecture

A target generated agent has a specific architecture divided into two parts, the *behavior part* and the *control part*. The behavior part includes the real agent behavior and strategies of regulation defined by the developer. Indeed, we would like not only that the verification fails when an inconsistency is detected, but that the agent regulates its behavior whenever a law is transgressed. The control part includes the set of Petri nets representing the laws associated with the agent and makes sure of the detection of the laws violation. The connections between the program and the models are effective thanks to a sending of information from the behavior part to the control part. To allow this sending of information, we instrument the behavior part by inserting automatically some control points associated with the events and states contained in the laws. The control part receives the information and verifies the respect of the laws.

4.3 The generation

The generation of a self-controlled agent comes down to the generation of the Petri nets representing the laws concerning the agent and the instrumentation of the agent behavior to detect the occurrence of events and states expressed in the laws.

4.3.1 The instrumentation

We propose an automatic instrumentation of the agent behavior program to monitor the occurrence of the events and states, defined in the laws, by means of **control points**. This instrumentation is done thanks to the hooks defined by the developer, between the concepts describing the model and its implementation. In order to do that, we draw our inspiration from the principle of weaving. The **weaving** is an important part of aspect programming [14]. The latter uses the weaving to inject aspects in classes of an application, at the methods level, to modify the system execution after the compilation. An aspect is a module representing crosscutting concerns. The interest of aspect programming to integrate the monitoring in an application

was demonstrated in another light by [8]. So, our approach could be summarize as follows:

1. Extracting the events or states to be detected.
2. For each event or state, searching for the hook that has been provided by the developer.
3. Injecting, before or after that hook, the piece of code allowing to send the information to the control part and to recover of possible information of transgression. The latest enables the agent to start a strategy of regulation.

4.3.2 The generation of Petri nets

The generation of a Petri net representing a law is divided into three stages:

1. The translation of the law in a logic expression L , in order to point out a set of atomic expressions, $\{e_1, \dots, e_n\}$.
2. Incrementally:
 - (a) The deduction of a set of Petri nets, $\{p_1, \dots, p_n\}$ representative of each expression in $\{e_1, \dots, e_n\}$.
 - (b) The merging of all the nets in $\{p_1, \dots, p_n\}$ from the relations between e_1, \dots, e_n to obtain a final Petri net, P , representing the law.

To perform the step (1), each operator in the language has a meaning in dynamic deontic logic. We use a set of translation rules to obtain the logic expression representing the law (cf. Set of Rules 1). For the step (2), we propose to represent atomic expression by means of a Petri net, as in the set of rules 2; and merging rules, as described in the set of rules 3. To express a prohibition we use an *inhibitor arc* and to express a time, we use a *temporal Petri net*. The final Petri net P representing the law, is embedded into the control part of each agent submitted to the law.

Set of Rules 1

- Let Act be a set of actions.
 Let $Assert$ be a set of assertions.
 Let $State$ be a set of states included in $Assert$.
 $\forall \alpha \in Act, \phi \in Assert, \beta \in State$.
- (1) $FORBIDDEN \alpha \equiv F\alpha$
 - (2) $OBLIGED \alpha \equiv O\alpha$
 - (3) $\phi AFTER \alpha \equiv [\alpha]\phi$
 - (4) $FORBIDDEN \alpha_1 BEFORE \alpha_2 \equiv done(\alpha_2) \vee F\alpha_1$
 - (5) $OBLIGED \alpha_1 BEFORE \alpha_2 \equiv \neg done(\alpha) \vee O\alpha_1$
 - (6) $\alpha_1 AND \alpha_2 \equiv (\alpha_1; \alpha_2) \cup (\alpha_2; \alpha_1)$
 - (7) $\alpha_1 THEN \alpha_2 \equiv \alpha_1; \alpha_2$
 - (8) $\phi IF \beta \equiv \beta \supset \phi$
 - (9) $FORBIDDEN \alpha - Sec \equiv done(time(Sec)) \vee F\alpha$
 - (10) $OBLIGED \alpha - Sec \equiv \neg done(time(Sec)) \vee O\alpha$
 - (11) $\phi + Sec \equiv [time(Sec)]\phi$

Set of Rules 2

- Let $PN = \langle P, T, Pre, Post \rangle$ be a Petri net, where:
 P : a set of places, $P = (p_1, p_2, \dots, p_n)$,
 T : a set of transitions, $T = (t_1, t_2, \dots, t_n)$,
 Pre : a function, $P \times T \rightarrow N$, which defines directed arcs from places to transitions. (Note Pre^* , when the directed arc is an inhibitor arc)
 $Post$: a function, $P \times T \rightarrow N$, which defines directed arcs from transitions to places
 t_α be the transition associated with the action α .
 In the following, the rule $X \Rightarrow Y$ means that X is a logical assertion and Y the corresponding Petri net.
 $\forall \alpha \in Act, \beta \in State$
- (1) $F\alpha \Rightarrow \langle (p_i, p_j), t_\alpha, Pre^*(p_i, t_\alpha), Post(p_j, t_\alpha) \rangle$
 - (2) $O\alpha \Rightarrow \langle (p_i, p_j), t_\alpha, Pre(p_i, t_\alpha), Post(p_j, t_\alpha) \rangle$
 - (3) $done\alpha \Rightarrow \langle (p_i, p_j), t_\alpha, Pre(p_i, t_\alpha), Post(p_j, t_\alpha) \rangle$

- (4) $\neg done\alpha \Rightarrow < p_i, p_j, t_\alpha, Pre^*(p_i, t_\alpha), Post(p_j, t_\alpha) >$
- (5) $\beta \Rightarrow < (p_i, \beta, p_j), t_i, t_j, (Pre(p_i, t_i), Pre(\beta, t_j)), (Post(\beta, t_i), Post(p_j, t_j)) >$
- (6) $\alpha \Rightarrow < (p_i, p_j), t_\alpha, Pre(p_i, t_\alpha), Post(p_j, t_\alpha) >$

Set of Rules 3

Let t_α be the transition associated with the event α .
 Let t_ϕ be the transition associated with the event used in ϕ .
 Let $Pre(p,t)$ be the function which returns the input place p of a transition t .
 Let $Post(p,t)$ be the function which returns the output place p of a transition t .
 Let $merge_p(p_1, p_2)$ be the operator of fusion of two places p_1 and p_2 .
 $\forall \alpha \in Act, \phi \in Assert, \beta \in State$
 (1) $[\alpha]\phi \Rightarrow merge_p(Pre(p, t_\alpha), Pre(p, t_\phi))$.
 (2) $\phi_1 \vee \phi_2 \Rightarrow merge_p(Pre(p, t_{\phi_1}), Pre(p, t_{\phi_2}))$.
 (3) $\phi_1 \wedge \phi_2 \Rightarrow separation$ in two Petri nets : PN_{ϕ_1} and PN_{ϕ_2}
 (4) $\beta \supset \phi \Rightarrow merge_p(Pre(p, t_\phi), \beta)$

5 The detection of transgression

We remind that the behavior part sends information about its states and events to the control part. From this information, the control part must verify if the laws associated with the agent are respected. To detect the laws violation, the control part uses the two following algorithms:

Algorithm 1 In the main part of the control part

- 1: Let I be an information about the agent behavior.
 - 2: Let $\{t_1, \dots, t_n\}$ be the set of transitions associated with I .
 - 3: Let $\{P_1, \dots, P_m\}$ be the set of Petri nets associated with the agent.
 - 4: Let $\{Pact_1, \dots, Pact_p\}$ be the set of activated Petri nets (i.e. associated with the laws to be manage)
 - 5: Let t_{ij} be the transition i of the net j .
 - 6: **for all** $P_k \in \{P_1, \dots, P_m\}$ with $t_{1k} \in \{t_1, \dots, t_n\}$ **do**
 - 7: $Pact_{p+1} \leftarrow$ create an instance of P_k
 - 8: add $Pact_{p+1}$ in $\{Pact_1, \dots, Pact_p\}$
 - 9: **end for**
 - 10: Let $\{Pact_1, \dots, Pact_l\}$ be the set of the activated Petri nets including a $t_{ij} \in \{t_1, \dots, t_n\}, j \in \{1, \dots, l\}$
 - 11: **for all** $Pact_j \in \{Pact_1, \dots, Pact_l\}$ **do**
 - 12: inform $Pact_j$ of the information associated with t_{ij}
 - 13: **end for**
-

Algorithm 2 Within the instances of Petri net

- 1: Let $\{t_1, t_n\}$ be the set of transitions of the Petri net.
 - 2: Let I be the sent information.
 - 3: Let t_I be the transition associated with the information $I \in t_1, \dots, t_n$.
 - 4: A transition t_i is **activated** if a token stands in all the previous places of t_i (in our Petri net the arcs are one-valuated).
 - 5: **if** t_I is activated **then**
 - 6: **if** t_I is fireable **then**
 - 7: fire the transition t_I
 - 8: **else**
 - 9: throw exception
 - 10: **end if**
 - 11: **end if**
-

The control part receives the information and generates a new instances of each Petri net beginning by the transition associated with the information. The monitoring of the law, i.e. the associated Petri net, is **activated**. Then, the control part forwards the information to all the instances where the information is expected. If the transition associated with the information is activated, the instance verifies if the transition is fireable (according to the line 6, algo 2) and changes the Petri net. When an inconsistency is detected, the instance throws an exception and the control part warns the behavior part by sending

information of transgression. An instance is destroyed when the net is in a final state, so when the law is respected or when the delay of the law observation is elapsed.

When the behavior part sends an information to the control part, it is stopped until the control part permits its restarting. The temporary deadlock is essential if we want to prevent the execution of forbidden actions.

6 About the multiagent behaviors

In the previous parts, we have presented the whole functioning of the self-controlled agents. Particularly, we tackle a single agent view, that is when a law affects only one agent. Even if this first part solves a set of problems related to individual agent behavior, we think that shifting to the multiagent behaviors rises other problems. Indeed, the question is the control of the multiagent system behavior when several agents, individually correct, are put together. A first step to answer this question is the handling of the control when a law is applied to several agents.

Our aim is to distribute as much as possible the control into each agent involved in the law. We would like to avoid a centralized solution. We emphasize three points to put forward in that case:

- How is the distribution of the Petri net done among all the agents concerned by the law?
- How do the control parts of the agents collaborate to detect a law transgression?
- How is the regulation done? Who is the culprit?

6.1 The net distribution

The Petri net representing a law applied to several agents is deduced as in a single agent context. Then, the net is distributed into the control parts of the agents concerned by the law. This distribution is done by following the algorithm 3.

Algorithm 3 Petri net Distribution

- 1: Let L be a law.
 - 2: Let P be the Petri net representing L .
 - 3: Let $Nbagent$ be the number of agents concerned by L .
 - 4: Let $\{T_{m1}, \dots, T_{mn}\}$ be the set of transitions where the information associated with t_{mi} comes from the agent $AG_m, m \in \{1, \dots, Nbagent\}$.
 - 5: Let $Ppre_{mt}$ be the input place of the transition t of the agent AG_m .
 - 6: Let $Ppost_{mt}$ be the output place of the transition t of the agent AG_m .
 - 7: **for all** $T \in \{T_{m1}, \dots, T_{mn}\}$ **do**
 - 8: put T in the control part of the agent AG_m .
 - 9: **end for**
 - 10: **if** $Ppost_{mt} \equiv Ppre_{(m+1)t'}$ **then**
 - 11: put $Ppost_{mt}$ in the control part of the agent $AG_{(m+1)}$.
 - 12: **end if**
 - 13: **if** $Ppre_{(m+1)t'} \equiv Ppre_{mt}$ **then**
 - 14: put $Ppre_{mt}$ in the control part of the agent AG_m .
 - 15: **end if**
 - 16: **for all** $t' \in \{1, \dots, n\}, m' \in \{1, \dots, Nbagent\}$ **do**
 - 17: **if** $Ppost_{mt} \neq Ppre_{m't'}$ **then**
 - 18: put $Ppost_{mt}$ in the control part of the agent AG_m .
 - 19: **end if**
 - 20: **if** $Ppre_{mt} \neq Ppost_{m't'}$ **then**
 - 21: put $Ppre_{mt}$ in the control part of the agent AG_m .
 - 22: **end if**
 - 23: **end for**
-

Let us note that the control parts are only linked through the arcs between places and transitions (themselves distributed over the control parts of agents). These links represent the information flow between the control parts (i.e. the flow of the token).

6.2 The collaboration

A control part is willing to listen to the behavior part of its agent and to the control part of the other agents. Indeed, when a law is distributed into several agents, the control parts of these agents are able to exchange information about the occurrence of events and states. We have seen that the net distribution between the control parts is in such a way that only branches $Transition \rightarrow Place$ or $Place \rightarrow Transition$ link every pieces of the net. These arcs represent the flow of an information between the agents. In order to explain the collaboration between the control parts, we propose to use an example.

So, when a control part, C_A , receives an information from its agent, if this information is associated with a transition T whose the next place is in the control part of another agent, C_B , then C_A sends information about the firing of this transition, (actually, it sends the token) to the control part C_B and waits for an acknowledgment of receipt. During this waiting, the execution of the agent is temporarily stopped and the information associated with T is considered as always available. The control part C_B receives the information, sends the acknowledgment to the control part C_A and executes the algorithm 1, as if the information comes to its associated behavior part. When C_A receives the acknowledgment, the transition can be really fired, the information associated with T is consumed and the agent behavior can continue.

6.3 The regulation

The detection of a law transgression generates an exception in the control part. This exception generates the transmission of information of transgression from the control part to the behavior part of the agent. In the context of a law applied to several agents, the mechanism remains the same. All the control parts detecting a transgression, send the exception to the associated behavior part. So, we suppose that the strategies of regulation solve the problem of who is the culprit. Putting such strategy in place in the agents is not trivial and will be treated in our future work.

7 Implementation

Our approach is implemented in a framework: SCAAR. This framework provides the means to describe laws, concepts... and to generate self-controlled agents. A prototype of SCAAR has been implemented in SICStus Prolog. This first version provides a part of the language, to describe single agent laws and the generation, from the laws description, of the Petri nets. The net distribution into several agents is in progress. The behavior of the control part has been implemented using the algorithms described in the section 5. The generated self-controlled agents are made up of two Prolog processes, one executing the real agent behavior, one executing the control part behavior. These two parts communicate by TCP/IP channel for the messages passing about the occurrence of events in the agent behavior and the detection of laws transgression.

Our approach has been applied to a multiagent application of mobile naval targets localization, developed in our service: *Interloc*. In Interloc, planes seek to detect boats, in a passive way. The system is implemented in Prolog. In order to demonstrate the robustness of MAS application, Interloc provides a set of wrong agent behaviors, like:

- Dumb: The agent sends no message.
- Deaf: The agent reads no message.

- Selfish: The agent always refuses the requested services.
- Absurd: The agent accepts the services but sends absurd results.
- Latecomer: The agent is always late replying.

Therefore, Interloc is an interesting testbed to validate our approach. We have expressed laws to detect wrong behaviors and the application has been executed with modified agents in order to detect these “undesirable behaviors”, by using the generator provided by SCAAR.

8 Conclusion

We have presented the principles of the autonomous agent control. Particularly, we have described the mechanisms and the architecture of the self-controlled agents, agents able to monitor their own behaviors. This monitoring is done from the specification of the laws that the agents must respect during their execution. A distributed solution to control multiagent behaviors has been proposed. An advantage of our approach is the possibility to define laws at a high level, allowing its application to several models of agents. Another point is the automatic generation of the agents and the simplification of the developer’s work to take the control into account. The framework SCAAR, allowing the generation of self-controlled agents has been introduced. Our future work is essentially to enhance the implementation and treat more carefully the regulation part.

REFERENCES

- [1] K.S. Barber and C.E. Martin, ‘Agent autonomy : Specification, measurement and dynamic adjustment’, in *Proc. of the Autonomy Control Software workshop at Autonomous Agents’99*, pp. 8–15, (May 1999).
- [2] C. Chopinaud, A. El Fallah Seghrouchni, and P. Taillibert, ‘Dynamic self-control of autonomous agents’, in *Post-Proceedings of PROMAS’05, LNAI 3862, Springer Verlag*, (2006).
- [3] E.M. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*, MIT Press, 2000.
- [4] M. de Sousa Dias and D.J. Richardson, ‘Issues on software monitoring’, Technical report, Department of Information and Computer Science, University of California, (July 2002).
- [5] M. Diaz, G. Juanole, and J.-P. Courtiat, ‘Observer-a concept for formal on-line validation of distributed systems’, *IEEE Trans. Softw. Eng.*, **20**(12), 900–913, (1994).
- [6] Staffan Hagg, ‘A sentinel approach to fault handling in multi-agent systems’, in *Revised Papers from the Second Australian Workshop on Distributed Artificial Intelligence: Multi-Agent Systems: Methodologies and Applications*, volume 1286 of *LNCS*, pp. 181–195. Springer-Verlag, (1996).
- [7] Mark Klein and Chrysanthos Dellarocas, ‘Exception handling in agent systems’, in *In proceedings of Agents’99*, pp. 62–68, (1999).
- [8] D. Mahrenholz, O. Spinczyk, and W. Schröder-Preikschat, ‘Program instrumentation for debugging and monitoring with AspectC++’, in *Proc. of the 5th IEEE International symposium on Object-Oriented Real-time Distributed Computing*, Washington DC, USA, (April 29 – May 1 2002).
- [9] M. Mansouri-Samani, *Monitoring of Distributed Systems*, Ph.D. dissertation, University of London, London, UK, 1995.
- [10] JJCH. Meyer, ‘A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic’, *Notre dame journal of formal logic*, **29**(1), 109–136, (Winter 1988).
- [11] T. Murata, ‘Petri nets: Properties, analysis and applications’, in *Proc. of IEEE*, **77**(4), 541–580, (April 1989).
- [12] Rodrigo Paes, gustavo Carvalho, carlos Lucena, Paulo Alencar, Hyggó Almeida, and Viviane Silva, ‘Specifying laws in open multi-agent systems’, in *ANIREM*, Utrecht, (July 2005).
- [13] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum, ‘Implementing norms in multiagent systems’, in *Proceedings of MATES’04*, Erfurt, Germany, (September, 29–30 2004).
- [14] Dean Wampler. The future of aspect oriented programming, 2003. White Paper, available on <http://www.aspectprogramming.com>.

Coalition Structure Generation in Task-Based Settings

Viet Dung Dang and Nicholas R. Jennings¹

Abstract. The coalition formation process, in which a number of independent, autonomous agents come together to act as a collective, is an important form of interaction in multi-agent systems. However, one of the main problems that hinders the wide spread adoption of coalition formation technologies is the computational complexity of *coalition structure generation*. That is, once a group of agents has been identified, how can it be partitioned in order to maximise the social payoff? To date, most work on this problem has concentrated on simple *characteristic function games*. However, this lacks the notion of tasks which makes it more difficult to apply it in many applications. Against this background, this paper studies coalition structure generation in a general task-based setting. Specifically, we show that this problem is NP-hard and that the minimum number of coalition structures that need to be searched through in order to establish a solution within a bound from the optimal is exponential to the number of agents. We then go onto develop an anytime algorithm that can establish a solution within a bound from the optimal with a minimal search and can reduce the bound further if time permits.

1 Introduction

The coming together of a number of distinct, autonomous agents in order to act as a coherent grouping is an important form of interaction in multi-agent systems. It has been long studied in game theory [3] and has recently become an important topic in multi-agent systems (where buyer agents may pool their requirements in order to obtain bigger group discounts), in grid computing (where multi-institution virtual organisations are viewed as being central to coordinated resource sharing and problem solving), and in e-business (where agile groupings of agents need to be formed in order to satisfy particular market niches). In all of these cases, the formation of coalitions aims to increase the agents' abilities to satisfy goals and to maximise their personal and/or the system's outcomes.

In this context, the coalition formation process can be viewed as being composed of three main activities [5]:

1. *Coalition structure generation*: forming coalitions of agents such that those within a coalition coordinate their activities, but those in different coalitions do not. This primarily involves partitioning the set of all agents in the system into exhaustive and disjoint coalitions². Such a partition is called a coalition structure. For example, in a multi-agent system composed of three agents $\{a_1, a_2, a_3\}$, there exist seven possible coalitions: $\{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}$ and five possible coalition structures: $\{\{a_1, a_2, a_3\}\}, \{\{a_1\},$

$\{a_2, a_3\}\}, \{\{a_2\}, \{a_3, a_1\}\}, \{\{a_3\}, \{a_1, a_2\}\}$ and $\{\{a_1\}, \{a_2\}, \{a_3\}\}$.

2. *Optimising the value of each coalition*: pooling the resources and tasks of the agents in a given coalition to maximise the coalition value. For example, given the coalition structure $\{\{a_1\}, \{a_2, a_3\}\}$, each of the two coalitions $\{a_1\}$ and $\{a_2, a_3\}$ will try to optimise its value.
3. *Payoff distribution*: dividing each coalition's value among its members. For example, if the coalition $\{a_2, a_3\}$ produces a payoff of X then this value needs to be divided between a_2 and a_3 according to some scheme (e.g. equality or stability).

Although these activities are distinct and, in a sense, conceptually sequential, it is also clear that they interact. For example, in a competitive environment, the coalition that an agent wants to join depends on the payoff that it is likely to receive (activities 1 and 3). However, in cooperative environments, where the agents work together to maximise the social welfare, payoff distribution is less important, and coalition structure generation that maximises the social welfare is the dominant concern. Here, our focus is on the first of these activities.

To date, most work on coalition structure generation has focused on simple *characteristic function games* (CFGs) (see section 6). In such settings, there is a value $v(S)$ for each and every subset S of A, known as the value of coalition S, which is the utility that members of S can jointly attain. However, in many practical applications, it is often the case that a coalition is formed in order to perform some task from a pool of potential tasks that the agent system has to perform [2] [6]. But in CFGs this connection between coalitions and tasks is absent; it is simply assumed that a coalition's value is attained when its agents pool their resources and tasks together somehow. The notion of tasks generally increases the complexity of the problem as there are typically many ways to map coalitions to tasks (as will be shown in more detail in section 2). Also, the assumption of CFGs that a coalition value is not affected by the actions of non-members is no longer valid, as a coalition's value now depends on the tasks that it performs and this set of tasks are accessible by all agents. Recently, there have been some work on such task-based coalition formations [2] [6]. However, the authors typically made some limiting assumptions about either the coalition values or the maximum size of a coalition (see section 6 for more details). In this paper we don't make such assumptions, as we deal with a general case.

Against this background, this paper advances the state-of-the-art in the following ways. First, we analyse the complexity of the coalition structure generation problem in task-based coalition formation settings. We show that the problem is NP-hard and, moreover, the minimum number of coalition structures that need to be searched through in order to establish a solution within a bound from the optimal is exponential to the number of agents. Second, we develop an anytime algorithm that can establish a solution within a bound from

¹ School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK, email: {vdd, nrj}@ecs.soton.ac.uk.

² Some research also considers non-disjoint coalitions (see section 5 for details).

the optimal and can reduce the bound further if time permits. We also consider a special case where each coalition carries out at most one task (referred as single-task coalitions in our paper) and show that in this case, the minimum number of coalition structures that need to be searched through in order to establish a bound from the optimal is much smaller than the general case.

The rest of the paper is structured as follows. Section 2 introduces our task-based coalition formation model. Section 3 analyses the complexity of the coalition structure generation problem and presents a minimal search to establish a bound from the optimal. Section 4 presents our anytime algorithm. Section 5 deals with the case of single-task coalitions. Finally, section 6 covers related work and section 7 concludes.

2 Task-Based Coalition Formation

This section formalises coalition structure generation in task-based settings, an extension of coalition formation in CFGs. Let A be the set of agents, and n be the number of agents in A (i.e. $|A| = n$). Let T be the set of tasks and m be the number of tasks: $T = \{t_1, t_2, \dots, t_m\}$. As an extension of coalition formation in CFGs, we consider settings where for each subset $C \subseteq A$ and each set of tasks $V \subseteq T$ there is a coalition value $v(C, V)$ for the coalition (C, V) . The coalition value $v(C, V)$ can be considered as the utility that agents in coalition C can attain by performing the tasks in V .³ Thus, while in traditional CFG the coalition value is a function $v : 2^A \rightarrow R$, in our case it is a function $v : 2^A \times 2^T \rightarrow R$. It can be seen that for each CFG coalition C there are $(2^m - 1)$ corresponding task-based coalitions (C, V) as there are $(2^m - 1)$ possible set of tasks. As there are $(2^n - 1)$ CFG coalitions C , there are $(2^n - 1)(2^m - 1)$ task-based coalitions.

As in [3], we assume that every coalition value is non-negative:

$$v(C, V) \geq 0, \forall C \subseteq A, V \subseteq T \quad (1)$$

Definition 1 A coalition structure CS is a partition of A into exhaustive disjoint (non-empty) coalitions, each of which will do a different set of tasks such as: i) these sets of tasks are disjoint, ii) at most one of them can be empty (that is, at most one coalition will do nothing, because if two or more of them do nothing then we can merge them together into one)⁴. Thus, $CS = \{(C_1, V_1), (C_2, V_2), \dots, (C_k, V_k)\}$, $1 \leq k \leq n$, coalition C_i , $1 \leq i \leq k$, will do the set of tasks V_i such that the following conditions are satisfied:

- $C_i \subseteq A, C_i \neq \emptyset, \forall 1 \leq i \leq k$
- $C_i \cap C_j = \emptyset, \forall 1 \leq i, j \leq k, i \neq j$
- $\bigcup_{i=1}^k C_i = A$
- $V_i \subseteq T, \forall 1 \leq i \leq k$
- $V_i \cap V_j = \emptyset, \forall 1 \leq i, j \leq k, i \neq j$
- $|\{V_i | V_i = \emptyset\}| \leq 1$

For example, for $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, the following is a coalition structure: $\{\{\{a_1, a_2\}, \{t_1\}\}, \{\{a_3\}, \{t_2, t_3\}\}, \{\{a_4, a_5\}, \{t_4, t_5\}\}\}$. In the above coalition structure, coalition $\{a_1, a_2\}$ does task t_1 , coalition $\{a_3\}$ does tasks t_2 and t_3 , coalition $\{a_4, a_5\}$ carries out tasks t_4 and t_5 , and agent a_6 does nothing (also task t_6 is not carried out by any coalition).

³ In this paper, we use the word *coalition* to refer to both a group of agents (e.g. coalition C) and a group of agents with a set of tasks to perform (e.g. coalition (C, V)).

⁴ The value of a coalition that does nothing is 0, that is, $v(C, \emptyset) = 0$.

Proposition 1 Let the size of a coalition structure be the number of coalitions that it contains. Then the size of any task-based coalition structure will be less than the number of tasks plus 1. That is, for any coalition structure $CS = \{(C_1, V_1), (C_2, V_2), \dots, (C_k, V_k)\}$, we have:

$$k \leq m + 1 \quad (2)$$

PROOF. From definition 1 we have: the sets V_i are disjoint and at most one of them is empty. Thus the union of the sets V_1, V_2, \dots, V_k must have at least $k - 1$ elements. But this union is a subset of T and T has m elements. Thus we have $k - 1 \leq m$ or $k \leq m + 1$. \square

Proposition 2 For each CFG coalition structure (C_1, C_2, \dots, C_k) ($k \leq m + 1$), there are $(k + 1)^m$ corresponding task-based coalition structures.

PROOF. For each task $t \in T$, there are $k + 1$ possibilities: t is carried out by a coalition C_i , $1 \leq i \leq k$ or is not carried out by any coalition. As there are m tasks, there are $(k + 1)^m$ combinations of possibilities to form a task-based coalition structure from that CFG coalition structure. \square

As in traditional CFG, we consider the *value* of a coalition structure $V(CS)$ in terms of its social welfare. That is, the value of a coalition structure is the sum of the values of its coalitions:

$$V(CS) = \sum_{i=1}^k v(C_i, V_i)$$

for $CS = \{(C_1, V_1), (C_2, V_2), \dots, (C_k, V_k)\}$.

Also let L be the set of all coalition structures.

Given the above terms, the problem of coalition structure generation for task-based coalition formation is then to find a coalition structure CS^* that maximises the social welfare. That is:

$$CS^* = \operatorname{argmax}_{CS \in L} V(CS)$$

Moreover, a solution CS' is said to be *within a bound b from the optimal solution* iff: $V(CS^*)/V(CS') \leq b$.

Having presented the settings in this section, the next section analyses the complexity of the problem.

3 Complexity Analysis

In this section, we analyse the complexity of the problem and discuss the smallest number of coalition structures that need to be searched through in order to establish a solution that is within a bound from the optimal one.

First, we investigate the size of the search space. Just as in the case of CFGs, the set of all coalition structures can be partitioned into a number of layers, each layer L_k , $1 \leq k \leq n$ holds all coalition structures with size k .

In CFG settings, the number of coalition structures in L_k is $S(n, k)$, widely known in Mathematics as *the Stirling number of the Second Kind* [4]. The value of $S(n, k)$ can be computed by the following standard formula:

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k-i)^n$$

In task-based settings, the number of coalition structures in L_k can be calculated as follows.

Proposition 3 *The number of coalition structures with size k ($1 \leq k \leq m+1$) can be computed by the following formula:*

$$|L_k| = (k+1)^m \cdot S(n, k)$$

PROOF. As for each CFG coalition structure (C_1, C_2, \dots, C_k) ($k \leq m+1$), there are $(k+1)^m$ corresponding task-based coalition structures (proposition 2) and there are $S(n, k)$ CFG coalition structures with size k , the number of coalition structures with size k is $(k+1)^m \cdot S(n, k)$. \square

Theorem 1 *The total number of coalition structures is $\sum_{k=1}^{\min(n, m+1)} (k+1)^m$.*

PROOF. As the number of coalition structures with size k ($1 \leq k \leq m+1$) is $(k+1)^m \cdot S(n, k)$ (proposition 3) and $k \leq n$ (as each coalition has at least one agent), thus the total number of coalition structures is $\sum_{k=1}^{\min(n, m+1)} (k+1)^m$. \square

With this knowledge, we now discuss how a bound from the optimal can be established while searching as little of the space as possible. In CFG settings, it has been shown that searching two CFG layers L_1 and L_2 is the quickest way to ensure a bound from the optimal [5] (quickest here means searching the smallest number of coalition structures). In our task-based settings, however, we will show that we only need to search a portion of the corresponding layers L_1 and L_2 in order to establish a bound.

Theorem 2 *The smallest number of coalition structures that need to be searched in order to establish a bound from the optimal is at least $(2^{m+n-1} - 1)$.*

PROOF. Suppose K^* is the smallest set of coalition structures that need to be searched in order to establish a bound from the optimal. That is, K^* contains the smallest number of coalition structures to be searched in order to establish a bound from the optimal. Because the value of a coalition $v(C, V)$ can be arbitrarily big, in order to establish a bound from the optimal, every coalition (C, V) ($V \neq \emptyset$) needs to be included in one of the coalition structures in K^* .

First, we can see that for any coalition (C, V) such that $C = A$ or $V = T$, the coalition structure that contains it cannot contain any other coalition, as there is either no agent or no task left. Thus the number of coalition structures that need to be searched through to cover these coalitions (and only them) is the number of these coalitions.

Now let us count the number of coalitions (C, V) such that $C = A$ or $V = T$. There are $(2^m - 1)$ coalitions (A, V) as there are $(2^m - 1)$ non-empty subsets of T . Similarly, there are $(2^n - 1)$ coalitions (C, T) . However, the coalition (A, T) is counted twice, so we have the number of coalitions (C, V) such that $C = A$ or $V = T$ is: $(2^m - 1) + (2^n - 1) - 1 = 2^m + 2^n - 3$. Thus the number of coalition structures that need to be searched through to cover these coalitions (and only them) is $2^m + 2^n - 3$.

Now, consider the coalitions (C, V) such that $C \subset A$ and $V \subset T$ (that is, C is a strict subset of A and V is a strict subset of T). Let us count the number of agent occurrences in every coalition⁵. For every

$1 \leq i \leq n-1$, as there are $\binom{n}{i}$ subsets⁶ of size i of A and $(2^m - 2)$ non-empty strict subsets of T , there are $\binom{n}{i} \cdot (2^m - 2)$ coalitions (C, V) where C is of size i .

Thus the number of agent occurrences in all coalitions (C, V) such that $C \subset A$ and $V \subset T$ is:

$$\sum_{i=1}^{n-1} (2^m - 2) \cdot \binom{n}{i} \cdot i = (2^m - 2) \cdot \sum_{i=1}^{n-1} \binom{n}{i} \cdot i \quad (3)$$

$$= (2^m - 2)n \cdot \sum_{i=1}^{n-1} C_{n-1}^{i-1} = (2^m - 2)n \cdot (2^{n-1} - 1) \quad (4)$$

As in each coalition structure, the number of agent occurrences is at most n , we need to search through at least $\frac{(2^m - 2)n \cdot (2^{n-1} - 1)}{n} = (2^m - 2) \cdot (2^{n-1} - 1)$ coalition structures to cover all coalitions (C, V) such that $C \subset A$ and $V \subset T$.

From the above results, we have the smallest number of coalition structures that need to be searched through in order to establish any bound from the optimal is $2^m + 2^n - 3 + (2^m - 2) \cdot (2^{n-1} - 1) = 2^{m+n-1} - 1$. \square

Theorem 1 trivially leads to the following corollary.

Corollary 1 *The problem of coalition structure generation for task-based settings is NP-hard.*

Now, the following question naturally arises: is there a set of exactly $2^{m+n-1} - 1$ coalition structures that after searching through all of its coalition structures, a bound from the optimal can be established? We will show that there does indeed exist such a set in the theorem below.

Theorem 3 *The smallest number of coalition structures that need to be searched in order to establish a bound from the optimal is $2^{m+n-1} - 1$.*

PROOF. As we have proved in Theorem 1 that we need to search at least $2^{m+n-1} - 1$ coalition structures in order to establish a bound from the optimal, we just need to show that there exists a set of $2^{m+n-1} - 1$ coalition structures that after searching through this set, a bound from the optimal will be established. Thus we will construct such set of coalition structures below.

Consider the following set of coalition structures: $K = K_1 \cup K_2$ where K_1 is the set of coalition structures $\{(A, V)\}$ in which V is a non-empty subset of T and K_2 is the set of coalition structures $\{(C, V), (A \setminus C, T \setminus V)\}$ where C is a strict subset of A and V is a subset of T .

First, we will show that the number of coalition structures in K is exactly $(2^{m+n-1} - 1)$. As there are $2^m - 1$ non-empty subsets of T , K_1 contains $2^m - 1$ coalition structures. For the set K_2 : there are $2^n - 2$ non-empty strict subsets C of A and 2^m subsets V of T , so there are $2^m(2^n - 2) = 2^{m+1}(2^{n-1} - 1)$ (C, V) coalitions. However, each coalition structure in K_2 contains two such (C, V) coalitions and each coalition (C, V) appears exactly once in a coalition structure in K_2 , thus the number of coalition structures in K_2 is $2^{m+1}(2^{n-1} - 1)/2 = 2^m(2^{n-1} - 1)$. Thus we can count the number of coalition structures by summing up the number of coalition structures in K_1 and K_2 together:

$$2^m - 1 + 2^m(2^{n-1} - 1) = 2^{m+n-1} - 1$$

⁵ When we count the number of agent occurrences, we take into account the repetition of agent occurrence. For example, in coalitions $\{a_1, a_2\}$ and $\{a_1, a_3\}$ the number of agents is 3, but the number of agent occurrences is 4.

⁶ $\binom{n}{k}$, or nC_k , is the standard mathematical notation to denote the number of different (unordered) combinations of k objects, taken from a pool of n objects and is calculated using the following formula: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Now we will show that after searching through all the coalition structures in K , a bound from the optimal can be established. Let $v(C^*, V^*) = \max_{C \subseteq A, V \subseteq T} v(C, V)$. Consider the following coalition structure: $CS^1 = \{(C^*, V^*)\}$ (if $C = A$) or $\{(C^*, V^*), (A \setminus C^*, T \setminus V^*)\}$ (if C is a strict subset of A), we then have $V(CS^1) \geq v(C^*, V^*)$ (according to assumption 1).

Now, for any coalition structure $CS = \{(C_1, V_1), (C_2, V_2), \dots, (C_k, V_k)\}$ we have:

$$V(CS) = \sum_{i=1}^k v(C_i, V_i) = \sum_{i=1, C_i \neq \emptyset, V_i \neq \emptyset}^k v(C_i, V_i)$$

Now, as there are n agents and m tasks, there are at most only $\min(m, n)$ coalitions with a non-zero value in any coalition structures. Thus:

$$V(CS) \leq \min(m, n) \cdot v(C^*, V^*) \leq \min(m, n) \cdot V(CS^1)$$

Thus $V(CS^*) \leq \min(m, n) \cdot V(CS^1)$. As $CS^1 \in K$, after searching through all the coalition structures in K , a bound $b = \min(m, n)$ from the optimal can be established. \square

Now, it can be seen that the set K contains most of layer L_1 (except $\{(A, \emptyset)\}$) and a subset of layer L_2 (as L_2 contains all coalition structures $\{(C, V_1), (A \setminus C, V_2)\}$ while K contains only coalition structures $\{(C, V_1), (A \setminus C, V_2)\}$ where $V_1 \cup V_2 = T$).

After establishing the bound $b = \min(m + 1, n)$, if additional time remains, it would be desirable to lower the bound with further search. Thus, the next section will present an anytime algorithm that can do this if time permits.

4 The Anytime Algorithm

In this section we present an *anytime algorithm* that can be interrupted at any time, and it establishes a monotonically improving bound. This algorithm is an adaptation of the anytime coalition structure generation algorithm in CFG settings developed in [1].

Definition 2 Let $SL(n, k, c)$ be the set of all coalition structures that have exactly k coalitions and at least one coalition that contains at least c agents.

Definition 3 Let $SL(n, c)$ be the set of all coalition structures whose cardinality is between 3 and $n - 1$ that have at least one coalition that contains at least c agents. That is:

$$SL(n, c) = \bigcup_{k=3}^{n-1} SL(n, k, c)$$

With these definitions in place, we can now express our algorithm for solving the problem (see Figure 1). Basically, at first it searches all the coalition structures in the set K . Then after that, our algorithm searches $SL(n, \lceil n(q-1)/q \rceil)$ with q running from $\lfloor \frac{\min(m,n)}{2} \rfloor$ down to 2. Note that we start from $q = \lfloor \frac{\min(m,n)}{2} \rfloor$ because we have shown that after searching through the set K a bound $b = \min(m, n)$ is established and, later in this section, we will show that after searching $SL(n, \lceil n(q-1)/q \rceil)$, our algorithm can establish a bound $b = 2q - 1$. Thus, we start from the biggest q such that $2q - 1 < \min(m, n)$ or $q = \lfloor \frac{\min(m,n)}{2} \rfloor$.

The next step is to show that the solution generated by the algorithm is within a bound from the optimal and that the bound is reduced further after each round. Thus ours is an **anytime algorithm**: it can be interrupted at any time and the bound keeps improving with an increase in execution time.

The algorithm proceeds as follows:

- Step 1: Search through the set K
- From step 2 onwards, search, consecutively, through the sets $SL(n, \lceil n(q-1)/q \rceil)$ with q running from $\lfloor \frac{\min(m,n)}{2} \rfloor$ down to 2. That is, search $SL(n, \lceil n(\frac{\min(m,n)}{2} - 1) / (\frac{\min(m,n)}{2}) \rceil)$ in step 2, $SL(n, \lceil n(\frac{\min(m,n)}{2} - 2) / (\frac{\min(m,n)}{2} - 1) \rceil)$ in step 3 and so on. Moreover, from step 3 onwards, as $SL(n, \lceil nq/(q+1) \rceil) \subseteq SL(n, \lceil n(q-1)/q \rceil)$, we only have to search through the set $SL(n, \lceil n(q-1)/q \rceil) \setminus SL(n, \lceil nq/(q+1) \rceil)$ in order to search through the set $SL(n, \lceil n(q-1)/q \rceil)$.
- At each step return the coalition structure with the biggest social welfare so far.

Figure 1. The coalition structure generation algorithm

Theorem 4 Immediately after finishing searching $SL(n, \lceil n(q-1)/q \rceil)$, the solution generated by our algorithm is within a finite bound $b = 2q - 1$ from the optimal.

PROOF. Due to the lack of space we omit the proof. It is, however, similar to the one in [1]. \square

5 Single-Task Coalitions

Up to now, we have considered the general setting in which a coalition can carry out an arbitrary number of tasks. However, in some settings, there is a natural restriction such that each coalition can only undertake a single task at any one time. Now, with this additional structure in place, we can improve upon the general results that we have derived so far.

In this special case, we consider settings where for each subset $C \subseteq A$ and each task t there is a coalition value $v(C, t)$ for the coalition (C, t) . Thus, here the coalition value is a function $v : 2^A \times T \rightarrow R$. It can be seen that for each CFG coalition C there are m corresponding single-task coalitions (C, t) as there are m possible tasks. As there are 2^n CFG coalitions C , there are $m2^n$ single-task coalitions.

A coalition structure CS is an exhaustive partition of A into $m+1$ disjoint (possibly empty) coalitions, each of the first m coalitions will do a different task and the last coalition does nothing. That is, $CS = \{(C_1, t_1), (C_2, t_2), \dots, (C_m, t_m), C_{m+1}\}$, coalition C_i , $1 \leq i \leq m$, will do task t_i , such as the following conditions are satisfied:

- $C_i \cap C_j = \emptyset$
- $\bigcup_{i=1}^{m+1} C_i = A$

For example, for $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $T = \{t_1, t_2, t_3\}$, the following is a coalition structure: $\{(\{a_1, a_2\}, t_1), (\{a_3\}, t_2), (\{a_4, a_5\}, t_3)\}$.

For the convenience of presentation, we will also write $CS = \{(C_1, t_1), (C_2, t_2), \dots, (C_m, t_m)\}$ or $CS = \{C_1, C_2, \dots, C_m\}$ in short.

Like the general case, the problem of coalition structure generation is then to find a coalition structure CS^* that maximises the social welfare (i.e. the sum of the values of all its coalitions).

By following a similar analysis as in the general case, we have the following corresponding results. First, we present the results regarding the size of the search space.

Proposition 4 For each CFG coalition structure (C_1, C_2, \dots, C_k) ($k \leq m+1$), there are $P(m+1, k)$ corresponding task-based coalition structures⁷

Theorem 5 The total number of coalition structures is $(m+1)^n$.

Definition 4 The size of a coalition structure is the number of non-empty coalitions that it contains.

Proposition 5 The number of coalition structures with size k ($1 \leq k \leq m+1$) can be computed by the following formula:

$$|L_k| = P(m+1, k)S(n, k)$$

We can see that the size of the search space in single-task coalitions case is much smaller than one in the general case. Thus, we can expect that the smallest number of coalition structures that need to be searched in order to establish a bound from the optimal will also be much smaller. This expectation is true, as will be shown below.

Theorem 6 The smallest number of coalition structures that need to be searched in order to establish a bound from the optimal is at least $m \cdot 2^{n-1}$.

Corollary 2 The problem of coalition structure generation for task-based coalition formation is NP-hard.

Theorem 7 The smallest number of coalition structures that need to be searched in order to establish a bound from the optimal is $m \cdot 2^{n-1}$.

In this case, the set of $m \cdot 2^{n-1}$ coalition structures that after searching through, a bound of the optimal can be established is:

$$K = K_1 \cup K_2 \cup \dots \cup K_m$$

with $K_i = \{\{C_1, C_2, \dots, C_m\}\}$ where for C is any subset of A :

- $C_j = \emptyset, \forall 1 \leq j < i$ or $j > i + 1$
- $C_i = C$
- $C_{i+1} = A \setminus C$ (for ease of presentation let $C_{m+1} \equiv C_1$)

Note that, for the above K_i , if a subset C of A is already considered, then we won't consider $A \setminus C$ as it will result in double counting. With this set K , a bound $b = \min(m, n)$ from the optimal can be guaranteed. As in the general case, after searching this set K , we can lower the bound if time permits by searching through the sets $SL(n, \lceil n(q-1)/q \rceil)$ with q running from $\lfloor \frac{\min(m,n)}{2} \rfloor$ down to 2.

6 Related Work

As introduced in section 1, to date, most work on coalition structure generation has focused on CFGs. In particular, Sandholm et al. [5] analysed the problem of coalition structure generation in CFG theoretically and proved that this problem is NP-hard and, moreover, even finding a sub-optimal solution requires searching an exponential number of solutions. They also developed an anytime algorithm that can establish a worst-case bound from the optimal. Dang and

⁷ $P(n, k)$, or ${}_nP_k$, is the standard mathematical notation to denote the number of different permutations of k objects, taken from a pool of n objects and is calculated using the following formula [7]: $P(n, k) = \frac{n!}{(n-k)!}$.

Jennings [1] later developed another anytime algorithm with worst-case bound guarantees that was significantly faster than Sandholm et al.'s algorithm when small bounds are desirable.

More specifically related to this work, Shehory and Kraus [6] consider a task-based setting where the coalitions can overlap. In their work, however, they reduce the complexity of the problem by limiting the size of the coalitions. They then develop a greedy algorithm that guarantees to produce a solution that is within a bound from the best solution possible given the limit on the number of agents. However, this best solution can be arbitrarily far from the actual optimal solution (without the limit on the size of the coalitions). Li and Sycara [2] addressed a similar task-based setting to our single-task based one. However they made some limiting assumptions (including the fact that the cost for a task only depends on the number of agents and the task cost per agent decreases as the number of agents increases). We place no such restrictions on our environment.

7 Conclusions and Future Work

In this paper, we examined the problem of coalition structure generation in task-based settings, which are an important extension over traditional CFGs. For this setting, we showed that the problem is NP-hard and, moreover, even a sub-optimal solution (which is within a finite bound from the optimal) requires searching an exponential number of coalition structures. We also developed an anytime algorithm that can establish a solution within a bound from the optimal with a minimal search and can reduce the bound further if time permits. We then considered the single-task coalitions case where each coalition can carry out at most one task and show that in this case, the minimum number of coalition structures that need to be searched through in order to establish a bound from the optimal is much smaller than the general case.

In future work, we intend to further reduce the complexity of the algorithm, especially if more assumptions can be made. For example, we may consider the case where the task-based coalition value is monotonic for each task, (that is, the larger number of agents in a coalition, the bigger its coalition value).

8 Acknowledgement

This research was undertaken as part of the ARGUS II DARP. This is a collaborative project involving BAE SYSTEMS, QinetiQ, Rolls-Royce, Oxford University and Southampton University, funded by the industrial partners together with the EPSRC, MoD and DTI.

REFERENCES

- [1] V. D. Dang and N. R. Jennings, 'Generating coalition structures with finite bound from the optimal guarantees', in *Proceedings of the Third International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 564–571, (2004).
- [2] C. Li and K. Sycara, 'A stable and efficient scheme for task allocation via agent coalition formation', *Algorithms for Cooperative Systems*, World Scientific, (2004).
- [3] A. Rapoport and J.P. Kahan, *Theories of Coalition Formation*, Lawrence Erlbaum Associates, 1984.
- [4] S. Roman, *The Umbral Calculus*, Academic Press, 1984.
- [5] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, 'Coalition structure generation with worst case guarantees', *Artificial Intelligence*, **111**(1-2), 209–238, (1999).
- [6] O. Shehory and S. Kraus, 'Methods for task allocation via agent coalition formation', *Artificial Intelligence*, **101**(1-2), 165–200, (1998).
- [7] J. V. Uspensky, *Introduction to Mathematical Probability*, New York: McGraw-Hill, 1937.

Programming Agents with Emotions

Mehdi Dastani and John-Jules Ch. Meyer¹

Abstract. This paper presents the syntax and semantics of a simplified version of a logic-based agent-oriented programming language to implement agents with emotions. Four types of emotions are distinguished: happiness, sadness, anger and fear. These emotions are defined relative to agent's goals and plans. The emotions result from the agent's deliberation process and influence the deliberation process. The semantics of each emotion type is incorporated in the transition semantics of the presented agent-oriented programming language.

1 Introduction

In the area of intelligent agent research many papers have been written on rational attitudes of agents pertaining to beliefs, desires and intentions (BDI) [8]. More recently one sees a growing interest in agents that display behaviour based on mental attitudes that go beyond rationality in the sense that also emotions are considered (e.g. [12]). There may be several reasons to endow artificial agents with emotional attitudes. For instance, one may be interested to imitate human behaviour in a more natural way [11]. This is in the first instance the perspective of a cognitive scientist, although, admittedly, also computer scientists involved in the construction of systems such as games or believable virtual characters may find this interesting. Our interest in emotional agents here is slightly different. We are interested in agents with emotions, since we believe that emotions are important heuristics that can be used to define effective and efficient decision-making process.

As Dennett [4] has pointed out, it may be sensible to describe the behaviour of complex intelligent systems as being generated by a consideration of their beliefs and desires (the so-called intentional stance). Bratman [1] has made a case for describing the behaviour of such systems in terms of their beliefs, desires and intentions. Here intentions play a role of stabilizing agents decisions, i.e., in deliberating about desires make choices and stick to these as long as it is 'rational'. Following this work it has been recognised that to design and create intelligent *artificial* agents this is a fruitful way of proceeding as well, leading to the so-called BDI architecture of intelligent agents [8]. Pursuing this line of developments we believe that it makes sense to go even further and incorporate also emotional attitudes in agent design and implementation. The role of emotions is to specify heuristics that make the decision process more efficient and effective and, moreover, result in more *believable* behavior [11].

If one takes the stance that endowing artificial agents with emotions is a worthwhile idea, the next question is how to construct or implement such agents. We believe that the agent architectures proposed in the literature are very complicated and hard to engineer, and this holds *a fortiori* for architectures that have been proposed for emotional agents, such as Sloman's CogAff architecture [10].

We start by looking at a logical description of agents with emotions as presented in [5]. Inspired by this logical specification, we aim at constructing agents with emotions through dedicated agent-oriented programming language, in which mental (BDI-like) notions have been incorporated in such a way that they have an unambiguous and rigorous semantics drawn from the meaning they have in the logical specification. In agent programming languages one programs the way the mental state of the agent evolves, and the idea goes back to Shoham [9], who proposed AGENT-0. *In particular, we devise a logic-based agent-oriented programming language, which is inspired by 3APL [3], and can be used to implement emotional agents, following the logical specification of [5] as closely as possible.*

The structure of this paper is as follows. In section 2, we briefly discuss four emotion types and explain their roles in agent programming. In section 3, the syntax of a simplified version a logic-based agent-oriented programming language is presented that allows the implementation of emotional agents. Section 4 presents the part of the semantics that is relevant for the involved emotion types. In section 5, the general structure of the deliberation process is presented. Finally, the paper will be concluded in section 6.

2 Emotions and Deliberation

Following [6, 7] we will incorporate four basic emotions: happiness, sadness, anger and fear. As in [5] we will concentrate on the functional aspects of these emotions with respect to the agent's *actions* ignoring all other aspects of emotions. We do this since we are (only) interested here in how emotions affect the agent's practical reasoning. Briefly, *happiness* results when in the pursuit of a goal subgoals have been achieved as a sign that the adopted plan for the goal is going well. In this case nothing special has to be done: plans and goals simply should persist. On the other hand, *sadness* occurs when subgoals are not being achieved. In this case we have to deliberate to either drop its goal or try to achieve it by an alternative plan. *Anger* is the result of being frustrated from not being able to perform the current plan, and causes the agent to try harder so that the the plan becomes achievable again. An agent of which a maintenance goal is being threatened becomes *fearful*, which causes it to see to it that the maintenance goal gets restored before proceeding with other activities.

From this description we see that the emotions that we will consider here are relativised with respect to certain parameters such as certain plans and goals. A consequence of this is that agent may at any time have many different emotions regarding different plans and goals. Thus in this paper we do not consider a kind of 'general' emotional state (e.g. happy) which is not further qualified. Another thing that should be clear is that some emotional types (such as happiness and sadness) come about from performing actions while other emotion types (such as anger and fear) come about from monitoring the

¹ Utrecht University, The Netherlands, Email: {mehdi,jj}@cs.uu.nl

deliberation process that decides the next action itself. The resulted emotions have an influence on this deliberation process. This is exactly what we are interested in here: how does these emotions arise and affect the agent's course of action.

3 Programming Language: Syntax

In this section, we propose a simplified version of a logic-based agent-oriented programming language that provides programming constructs for implementing agents with emotions. This programming language treats an agent's mental attitudes such as beliefs, goals, and plans, as data structures which can be manipulated by meta operations that constitute the so-called deliberation process. In particular, beliefs and goals are represented as formulas in data bases, plans consists of actions composed by operators such as sequence and iterations, and reasoning rules specify which goals can be achieved by which plans and how plans can be modified. The deliberation process, which constitutes the agent's reasoning engine, is a cyclic process that iterates the sense-reason-act operations. The deliberation process can be implemented by a deliberation language [2]. It should be clear that the deliberation language can be considered as a part of the agent-programming language.

In order to focus on different types of emotions and their computational semantics, we ignore many details that may be relevant or even necessary for a practical logic-based agent-oriented programming language. The practical extension of this logic-based programming language for agents without emotions is presented in [3].

Although the implemented agents will be endowed with emotions, the programming language does not provide any programming constructs to implement emotions. In fact, the syntax of the programming language is similar to the syntax of the language presented in [3]. However, the semantics of the language is extended with the emotional state. The idea is that the emotional state of an agent, which is determined by its deliberation process and the effects of the executed actions, influences the deliberation process itself. Thus, while the syntax of the programming language is similar to the one presented in [3], the semantics of this language is extended with an emotional state.

For this logic-based agent-oriented programming language, we assume a propositional language L , called the *base language*, and the propositional entailment relation \models . The beliefs and goals of an agent are propositional formulas from the base language L . In this paper, we distinguish two different goals types: achievement goals and maintenance goals. An achievement goal denotes a state that the agent wants to achieve and will be dropped as soon as the state is (believed to be) achieved. A maintenance goal denotes a state that the agent wants to maintain. In contrast to achievement goals, a maintenance goal can hold even if the state denoted by it is believed to hold.

The plans of an agent are considered as a sequence of basic actions and test actions. Basic actions are specified in terms of pre- and post-condition. They are assumed to be deterministic in the sense that they either fail (if the precondition does not hold) or have unique resulting states. The precondition of a basic action indicates the condition under which the action can be performed and the postcondition of a basic action indicates the expected effect of the action. The test actions check if propositions are believed, i.e., if propositions are derivable from the agent's belief base.

Definition 1 (plan) Let $Action$ with typical element α be the set of basic actions and $\phi \in L$. The set of plans $Plans$ with typical element

π is then defined as follows:

$$\pi ::= \alpha \mid \phi? \mid \pi_1; \pi_2$$

We use ϵ to denote the empty plan.

Planning rules are used for selecting an appropriate plan for a goal under a certain belief condition. A planning rule is of the form $\beta, \kappa \Rightarrow \pi$ and indicates that it is appropriate to decide plan π to achieve the goal κ , if the agent believes β . In order to check whether an agent has a certain belief or goal, we use propositional formulas from the base language L .

Definition 2 (plan selection rule) The set of plan selection rules \mathcal{R}_{PG} is a finite set defined as follows:

$$\mathcal{R}_{PG} = \{\beta, \kappa \Rightarrow \pi \mid \beta, \kappa \in L, \pi \in Plans\}$$

In the following, we use $belief(r)$, $goal(r)$, and $plan(r)$ to indicate, respectively, the belief condition β , the goal condition κ , and the plan π that occur in the planning rule $r = (\beta, \kappa \Rightarrow \pi)$.

Given these languages, an agent can be implemented by programming two sets of propositional formulas (representing the agent's beliefs and goals), a set of basic actions specified in terms of their pre- and postcondition, one set of planning rules, and an order on the set of planning rules to indicate the order according to which the planning rules should be applied.

Definition 3 (agent program) An agent program is a tuple $(\sigma, (\gamma_a, \gamma_m), A, PG, <)$ where $\sigma, \gamma_a, \gamma_m \subseteq L$, $A \subseteq Action$, $PG \subseteq \mathcal{R}_{PG}$, and $<$ is a strict order on PG .

For technical reasons, we demand that if two planning rules in PG have equivalent goal formulas in their heads, then they will be identical formulas. Moreover, we assume that an agent's plans are generated by applying planning rules and that an agent does not have initial plans. This simplification is due to the focus of this paper and can be relaxed for a practical agent-oriented programming language.

4 Programming Language: Semantics

The semantics of the logic-based agent-oriented programming language is defined by means of a transition system. A transition system for a programming language consists of a set of axioms and derivation rules for deriving transitions for this language. A transition is a transformation of one configuration into another and it corresponds to a single computation step. A configuration represents the state of an agent at each point during computation.

For the purpose of this paper, a configuration consists of a belief base σ representing the agent's beliefs, a goal base $\gamma = (\gamma_a, \gamma_m)$ representing the agent's (achievement and maintenance) goals, an action base A consisting of the specifications of basic actions, a plan base Π containing the plans, a set of planning rules PG , an ordering $<$ on the set of planning rules, and an emotion state consisting of emotions of the agent with respect to certain goals and/or plans.

Definition 4 (agent configuration) Let $\Sigma = \{\sigma \mid \sigma \subseteq L, \sigma \neq \perp\}$. An agent configuration is a tuple $(\sigma, (\gamma_a, \gamma_m), A, \Pi, PG, <, E)$ where $\sigma \in \Sigma$ is the belief base, $\gamma = (\gamma_a, \gamma_m)$ is the goal base, A is the action base, $\Pi \subseteq (L \times Plans)$ is the plan base, $PG \subseteq \mathcal{R}_{PG}$ is a set of planning rules, $<$ is a strict order on PG , and $E = (E_h, E_s, E_a, E_f)$ are emotion bases for happiness, sadness, anger, and fear, where $E_h \subseteq \{\text{happy}(\pi, \kappa, \kappa') \mid \pi \in Plans \& \kappa, \kappa' \in L\}$, $E_s \subseteq \{\text{sad}(\pi, \kappa) \mid \pi \in Plans \& \kappa \in L\}$, $E_a \subseteq \{\text{angry}(\pi) \mid \pi \in Plans\}$, and $E_f \subseteq \{\text{fearful}(\kappa) \mid \kappa \in L\}$.

Note that the elements of the plan base are defined as tuples consisting of a plan and a goal formula, which indicates the original intended state to be reached/maintained by the plan. Note also that the emotions of an agent are with respect to particular goals and plans such that, for example, an agent can be happy with respect to one of its goal/plan while it is sad with respect to another goal/plan. In the sequel, we often omit the set of basic actions, the set of planning rules PG and the ordering $<$ in the agent configuration for reasons of presentation, i.e., we use configurations of the form $\langle \sigma, \gamma, \Pi, E \rangle$ instead of $\langle \sigma, \gamma, A, \Pi, PG, <, E \rangle$. This is not problematic, since the set of basic actions, the set of planning rules, and the ordering defined on them does not change during executions of an agent.

Moreover, in order to check if an agent, which is represented by a configuration $\langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle$, has certain beliefs and (achievement or maintenance) goals, we use \models_b and \models_g defined as follows:

$$\begin{aligned} \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_b \phi &\Leftrightarrow \sigma \models \phi, \\ \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_g \text{achieve}(\kappa) &\Leftrightarrow \gamma_a \models \kappa, \\ \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_g \text{maintain}(\kappa) &\Leftrightarrow \gamma_m \models \kappa. \end{aligned}$$

In the following, we do not present the complete operational semantics of the proposed agent programming language, but provide only the transition rules that are related to the emotional states of the agents. Other transition rules for this agent programming language are trivial extensions of the operational semantics presented in [3].

For the transition rules, we assume a belief update operator $\tau : \Sigma \times Action \rightarrow \Sigma$ that determines the updates of the belief base by a basic action. Moreover, we assume $PreCond : Plans \rightarrow L$ and $PostCond : Action \rightarrow L$ to be functions that determine the precondition and the expected effect (postcondition) of a basic action, respectively. The precondition of a plan can be determined recursively in terms of the involved basic actions in a STRIPS like fashion. Moreover, we assume that the postcondition of an action is not necessarily derivable from the update of the belief base by the action, i.e., it is not generally the case that $\tau(\sigma, \alpha) \models PostCond(\alpha)$. This means that the expected effect of an action is not necessarily realized by the action execution (due to the unpredictability of the environment).

Before presenting the emotion related transitions, we need to present some preliminary concepts. We will do this in a rather informal way and refer to [5] for a rigorous treatment. First of all, we use dynamic logic expressions to specify the effects of actions: $[\alpha]\psi$ denotes that after all possible executions of action α it holds that ψ . $\langle\alpha\rangle\psi = \neg[\alpha]\neg\psi$ expresses there is an execution of α resulting in a state where ψ holds. Note that $\langle\alpha\rangle\top$ (α is executable) is equivalent with $PreCond(\alpha)$, and that $\langle\alpha\rangle\psi \rightarrow \langle\alpha\rangle\top$ is valid in dynamic logic. Furthermore, we have the validity $\langle\alpha; \pi\rangle\psi \leftrightarrow \langle\alpha\rangle(\langle\pi\rangle\psi)$.

As mentioned, a planning rule is meant to indicate which plan can be executed to realize a goal. A planning rule $r = (\kappa, \beta \Rightarrow \pi)$ is called correct, denoted by $correct(r)$, if the goal κ is achievable by the execution of the plan π , i.e., $correct(\kappa, \beta \Rightarrow \pi) \Leftrightarrow (\beta \rightarrow \langle\pi\rangle\kappa)$ where $\langle\pi\rangle\kappa$ states that (under condition β) after the performance of π , the state denoted by κ holds. We assume that the agent programmer is responsible for the correctness of the planning rules.

Moreover, given the agent configuration C , an agent *can* perform a plan π to achieve a goal κ if and only if π is executable (i.e., its precondition is derivable from the agent's belief base) and achieves κ after it is executed, i.e., $C \models Can(\pi, \kappa) \Leftrightarrow C \models_b \langle\pi\rangle\kappa$. (Note that $\langle\pi\rangle\kappa \rightarrow PreCond(\pi)!$ ²). The notion of 'Can' can be made more concrete in the present setting involving planning rules as follows:

² In [5] in the 'Can' operator also a notion of ability regarding plan π is incorporated, which may be viewed here as the agent's having access to all basic actions involved in the plan. For simplicity we omit this here.

An agent can (Can') perform a plan π to achieve a goal κ if and only if there exists a correct planning rule $r = (\kappa, \beta \Rightarrow \pi'; \pi)$, and the belief condition β is entailed by the agent's beliefs, i.e.,

$$\begin{aligned} C \models Can'(\pi, \kappa) \Leftrightarrow \exists r \in PG : & correct(r) \& \\ & goal(r) = \kappa \& \\ & belief(r) = \beta \& \\ & \exists \pi' \in Plans : plan(r) = \pi'; \pi \& \\ & C \models_b belief(r) \end{aligned}$$

Note that $Can'(\pi, \kappa) \Rightarrow Can(\pi, \kappa)$. Finally, as in [5] we employ a notion of possible intention to perform a plan π to achieve a goal κ stating that the agent 'Can' do π to achieve its goal κ . In the setting here it is operationalised as follows. An agent possibly intends to perform a plan π to achieve a goal κ if and only if the goal κ is a goal of the agent, and the agent can perform π to achieve κ , i.e., $C \models PossIntend(\pi, \kappa) \Leftrightarrow C \models Can'(\pi, \kappa) \& C \models_g achieve(\kappa)$.

Note that the intended plans are assumed to be generated by the applications of planning rules. Moreover, if the first part π' of an intended plan $\pi'; \pi$ gets executed, the agent remains intended to perform the rest of the plan (i.e., π) to achieve the original goal κ .

4.1 Happiness

According to [5], an agent that is happy observes that its subgoals are being achieved. In particular, an agent that has the intention to do π for achieving goal κ (denoted $I(\pi, \kappa)$ below), and is committed to it, and that believes that by performing the initial part α the subgoal κ' should be achieved, is *happy* (with respect to the remainder $\pi \setminus \alpha$ of the plan – to which it is still committed, the goal κ and subgoal κ') if after the performance of α it believes that indeed the subgoal κ' has been achieved. This is formulated as follows:

$$\begin{aligned} I(\pi, \kappa) \wedge Com(\pi) \wedge \alpha \preceq \pi \wedge B([\alpha]\kappa') \rightarrow \\ [\alpha]((B\kappa' \wedge Com(\pi \setminus \alpha)) \rightarrow \text{happy}(\pi \setminus \alpha, \kappa, \kappa')) \end{aligned}$$

Moreover, it is defined: $\text{happy}(\pi, \kappa, \kappa') \Leftrightarrow \text{happy}(\pi, \kappa)$ for all subgoals κ' that are deemed important/crucial by the agent. In this work, we assume that all subgoals are crucial to the agent.

We first observe that an agent becomes happy after the execution of actions. Therefore, the transition rule for action execution is an appropriate transition rule through which agents can become happy with respect to a plan and its corresponding goal and subgoal. In order to define the transition rule for action execution, we translate $I(\pi, \kappa)$ as possible intention $PossIntend(\alpha; \pi, \kappa)$, $Com(\pi)$ as the fact that the agent has the plan π in the plan base, and $B([\alpha]\kappa')$ as the fact that the postcondition of the basic action α is κ' . After the execution of the basic action α , if the updated belief base entails the postcondition of the basic action, then the agent becomes happy. Note that the agent becomes committed to the rest-plan π' of the plan $\pi = \alpha; \pi'$ since π' is added to the new plan base Π' . The transition rule for action execution can be defined as follows:

$$\frac{(\sigma, \gamma, \Pi, E) \models PossIntend(\alpha; \pi', \kappa) \& \\ (\alpha; \pi', \kappa) \in \Pi \& PostCond(\alpha) = \kappa' \& \tau(\sigma, \alpha) = \sigma'}{(\sigma, \gamma, \Pi, E) \rightarrow (\sigma', \gamma, \Pi', E')}$$

where

$$\Pi' = (\Pi \setminus \{(\alpha; \pi', \kappa)\}) \cup \{(\pi', \kappa)\}$$

$$E = (E_h, E_s, E_a, E_f)$$

$$\begin{aligned} E' = (E_h \cup \{\text{happy}(\pi, \kappa, \kappa')\}, E_s, E_a, E_f) \text{ if } \sigma' \models \kappa' \\ = E \text{ otherwise} \end{aligned}$$

According to [5], happiness causes a kind of persistence with respect to possible intention and commitments, i.e.,

$$I(\pi, \kappa) \wedge Com(\pi) \wedge \text{happy}(\pi, \kappa) \rightarrow [\text{deliberate}]I(\pi, \kappa) \wedge Com(\pi)$$

This means that intentions and commitments of an agent with respect to plans and goals persist through the deliberation *operations* (i.e., operations for selecting and applying reasoning rules) when the agent is happy with respect to those goals and plans. In order to ensure the persistence of intentions and commitments through the deliberation operations, we define in section 5 a deliberation *process* that does not drop any intention with respect to which the agent is happy. We assume that the deliberation operation, represented as *[deliberate]*, is a part of the deliberation process, represented as *[deliberate; executePlans]*. In fact, the deliberation process consists of deliberation operations plus the execution of plans (not a deliberation operation).

4.2 Sadness

A sad agent is disappointed about the way its plans are progressing, and will look for ways of revising its plans (or perhaps even adjust the goals to be achieved) and make them more realistic. In particular, an agent, who intends to perform a plan to achieve a goal and believes that the first action of its plan will have a certain effect, will become sad with respect to the rest of that plan after executing the first action if the agent believes that the expected effect is not achieved. This is formally specified as follows.

$$\begin{aligned} I(\pi, \kappa) \wedge Com(\pi) \wedge \alpha \preceq \pi \wedge \mathbf{B}([\alpha]\kappa') \rightarrow \\ [\alpha]((\mathbf{B}\neg\kappa' \wedge \mathbf{Com}(\pi \setminus \alpha)) \rightarrow \mathbf{sad}(\pi \setminus \alpha, \kappa)) \end{aligned}$$

Like happiness, we observe that an agent can become sad after executing a basic action, and that the transition rule for action execution is therefore an appropriate transition rule through which agents can become sad with respect to a plan and its corresponding goal. Under similar translation of logical concepts, the transition semantics for the sadness is as follows.

$$\frac{\langle \sigma, \gamma, \Pi, E \rangle \models PossIntend(\alpha; \pi', \kappa) \wedge (\alpha; \pi', \kappa) \in \Pi \wedge PostCond(\alpha) = \kappa' \wedge \tau(\sigma, \alpha) = \sigma'}{\langle \sigma, \gamma, \Pi, E \rangle \rightarrow \langle \sigma', \gamma, \Pi', E' \rangle}$$

where

$$\begin{aligned} \Pi' &= (\Pi \setminus \{(\alpha; \pi', \kappa)\}) \cup \{(\pi', \kappa)\} \\ E &= (E_h, E_s, E_a, E_f) \\ E' &= (E_h, E_s \cup \{\mathbf{sad}(\pi, \kappa)\}, E_a, E_f) \text{ if } \sigma' \not\models \kappa' \\ &= E \text{ otherwise} \end{aligned}$$

Sadness results in a revision of intention/plan or goal. In particular, an agent who is sad with respect to a plan that he intends to perform to achieve a goal, will become committed to either perform the plan if he can perform it, or otherwise generate an alternative plan and performs the new plan. This effect of sadness is specified as follows.

$$\begin{aligned} I(\pi, \kappa) \wedge Com(\pi) \wedge \mathbf{sad}(\pi, \kappa) \rightarrow \\ [deliberate]\neg I(\pi, \kappa) \vee \neg Com(\pi) \vee \\ Com(\text{if } Can(\pi, \kappa) \text{ then } \pi \text{ else replan}(\pi', \kappa); \pi') \end{aligned}$$

Observe that after *[deliberate]*, it holds:

$$I(\pi, \kappa) \wedge Com(\pi) \rightarrow \\ Com(\text{if } Can(\pi, \kappa) \text{ then } \pi \text{ else replan}(\pi', \kappa); \pi')$$

In order to operationalize this, we check after the deliberation operation if the agent still intends to perform (and is committed to) the plans with respect to which the agent is sad. If so, then we should ensure that the agent checks if he 'can' perform the plan before actually executing it; if he 'can not' perform it, then the agent should generate and execute an alternative plan. In order to simplify this, we check after the deliberation operation and after selecting a plan to execute, if the agent is sad with respect to the selected plan (obviously, the agent intend and is committed to the selected plan for execution) and if he still 'can not' perform it (if the plan 'can' be performed, then it will be executed). In such a case, an alternative plan will be generated

and executed. Given π as the selected plan to execute, the following deliberation step should be included in the deliberation process.

```
If  $\mathbf{sad}(\pi, \kappa) \in E_s \wedge C \not\models Can(\pi, \kappa)$  then
  {replan( $\pi'$ ,  $\kappa$ ); execute( $\pi'$ )}
else execute( $\pi$ )
```

We have assumed that the agent can generate an alternative plan by performing the deliberation operation *replan*(π' , κ), which provides an alternative plan π' to achieve the goal κ . If there is no alternative plan possible, this operation provides a plan which has already been generated. Details about this operation can be found in [2].

4.3 Anger

An agent gets angry if its plan is frustrated. This is specified as follows: $Com(\pi) \wedge \neg Can(\pi, \top) \rightarrow \mathbf{angry}(\pi)$

For the transition semantics, an active plan is considered to be frustrated if its precondition does not hold. This implies that the anger of an agent cannot be incorporated in the transition rule for action execution. We introduce an additional transition rule to update the angry state of the agent with respect to the plans that are not executable. This transition rule is specified as follows:

$$\frac{(\kappa, \pi) \in \Pi \wedge \langle \sigma, \gamma, \Pi, E \rangle \not\models Can(\pi, \top)}{\langle \sigma, \gamma, \Pi, E \rangle \rightarrow \langle \sigma, \gamma, \Pi, E' \rangle}$$

where

$$\begin{aligned} (\kappa, \pi) &\in \Pi \\ E &= (E_h, E_s, E_a, E_f) \\ E' &= (E_h, E_s, E_a \cup \{\mathbf{angry}(\pi)\}, E_f) \end{aligned}$$

Note that in our present setting we have that $Can(\pi, \top) \leftrightarrow PreCond(\pi)$. So, $Can(\pi, \top)$ is not derivable from an agent's configuration iff the precondition of the plan π does not hold. Note that we should ensure that this transition rule will be applied during each deliberation cycle.

An angry agent will see to it that he *will* be able to achieve its plans and goals. The effect of being angry is specified as follows:

$$\mathbf{angry}(\pi) \rightarrow [deliberate]Com(stit(PreCond(\pi)))$$

The specification of the effect of being angry indicates that the agent should try to realize the precondition of the plan with respect to which the agent is angry. This can be done by adopting the precondition of the plan as a goal. In order to realize this effect, the deliberation process will include the following deliberation step.

```
For all  $\mathbf{angry}(\pi) \in E_a$  do
  if  $C \not\models_g achieve(PreCond(\pi))$  then
    adopt\_goal(PreCond(\pi))
```

This deliberation step ensures that the agent will try to realize the conditions that enable the execution of plans with respect to which the agent is angry.

4.4 Fear

An agent becomes fearful if one of its maintenance goals is threatened. This is logically specified as:

$$\models \kappa \rightarrow \neg \kappa' \Rightarrow (goal_m(\kappa') \wedge G(\kappa)) \rightarrow \mathbf{fearful}(\kappa')$$

For the transition semantics, it means that an agent becomes fearful with respect to a goal if a transition takes place through which a goal is adopted that contradicts a maintenance goal. The transition semantics for the fearfulness is as follows.

$$\begin{aligned} \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \rightarrow \langle \sigma', (\gamma'_a, \gamma'_m), \Pi', E' \rangle \wedge \\ \gamma'_a \models \kappa \wedge \gamma'_m \models \kappa' \wedge \models \kappa \rightarrow \neg \kappa' \\ \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \rightarrow \langle \sigma', (\gamma'_a, \gamma'_m), \Pi', E'' \rangle \end{aligned}$$

where

$$\begin{aligned} E' &= (E'_h, E'_s, E'_a, E'_f) \\ E'' &= (E'_h, E'_s, E'_a, E'_f \cup \{\text{fearful}(\kappa')\}) \end{aligned}$$

This transition rule states that whenever there is a transition (action execution transition, rule application transition, etc.) possible through which the agent adopts a goal that threaten one of its maintenance goals, then the agent becomes fearful with respect to the maintenance goal.

The effect of the fearfulness is that the deliberation process should ensure that the feared maintenance goal holds before executing any plan. This is logically specified as follows.

$$\begin{aligned} \text{Goal}_m(\kappa') \wedge \text{Com}(\pi) \wedge \text{fearful}(\kappa') \rightarrow \\ [\text{deliberate}] \text{Com}(\text{if } \kappa' \text{ then } \pi \text{ else } \text{stit}(\kappa'); \pi) \end{aligned}$$

Normally, the deliberation *process* of an agent selects a plan and executes it. In order to ensure the effect of the fearfulness, the deliberation process will adopt the feared maintenance goals (such that the agent generate plans to realize them) before it proceeds with the execution of its selected plan. Below we make sure that the agent proceeds with executing its selected plan after it has adopted its feared goals. Suppose that the deliberation process has the following deliberation operation through which a plan is selected and executed.

$$\pi := \text{SelectPlanToExecute}(\Pi)$$

The effect of the fearfulness can be incorporated in the deliberation *process* by replacing this deliberation *operation* with the following sequence of deliberation operations.

$$\pi := \text{SelectPlanToExecute}(\Pi);$$

for all $\text{fearful}(\kappa') \in E_f$ such that $C \not\models_b \kappa'$ and
 $C \not\models_g \text{achieve}(\kappa')$ do

$$\{\pi := \text{adopt_goal}(\kappa'); \pi\};$$

Note that this for-loop generates a plan that consists of the selected plan preceded by the operations to adopt all feared maintenance goals. Note also that the effect of the fearfulness can be strengthened by adding the condition that the maintain goal should hold before proceeding with the executions of the selected plan. This can be done by replacing the body of the for-loop with the following statement:

$$\pi := \text{adopt_goal}(\kappa'); \pi'; \pi$$

5 Deliberation Process

For the purpose of this paper, we present the following general structure of the deliberation *process* for agents with emotional state.

```

While TRUE Do {
    Sensedata := perceive(environment);
    BeliefBase := update(BeliefBase, Sensedata);
    r := SelectPlanningRule(PG, <);
    ApplyPlanningRule(r);
    For all angry( $\pi_a$ )  $\in E_a$  do
        if  $C \not\models_g \text{achieve}(\text{PreCond}(\pi_a))$  then
            adopt_goal(PreCond( $\pi_a$ ));
     $\pi := \text{SelectPlanToExecute}(\Pi)$ ;
    If sad( $\pi, \kappa$ )  $\in E_s$  &  $C \not\models \text{Can}(\pi, \kappa)$  then
        {replan( $\pi', \kappa$ );  $\pi := \pi'$ }
    for all  $\text{fearful}(\kappa') \in E_f$  such that
         $C \not\models_b \kappa'$  and  $C \not\models_g \text{achieve}(\kappa')$  do
             $\{\pi := \text{adopt\_goal}(\kappa'); \pi\};$ 
            execute( $\pi$ );
    Apply transition rules for angry and fearful
}

```

The deliberation process is assumed to be an cyclic process consisting of perception, reason, and act cycles. During each cycle the

agent perceives its environment and updates its belief base accordingly, applies planning rules to generate plans for its goals, reasons about its emotions and realizes the effect of the emotions, selects and executes plans, and finally makes sure that the transition rules related to different types of emotions are applied. This deliberation cycle can be summarized as follows. Note that the transition rules for **angry** and **fearful** are applied during the last deliberation operation. The transition rules for **happy** and **sad** are applied by the statement **Execute** since these transition rules are the transitions for the execution of basic actions. Note also that the deliberation operations (i.e., the deliberation process except its execution part) does not drop any intention and commitment. Therefore, they ensure that all intentions and commitments, with respect to which the agent is happy, persist during the deliberation operation.

6 Conclusion

In this paper, we presented a logic-based agent-oriented programming language that allows the implementation of emotional agents. Four emotion types are discussed and their computational semantics incorporated in the transition system of the programming language. For each emotion type, we presented a transition rule that generates that specific emotions. Based on generated emotions, the deliberation process determines the effect of those emotion on the mental state of the agent. We aim to extend the set of emotion types in the future research. Moreover, we are working on an implementation of an interpreter for the presented programming language. The specified semantics of the emotion types can then be evaluated based on implemented agents. In particular we will test our concept of emotional agents in the realization of a companion robot for young children.

REFERENCES

- [1] M.E. Bratman, *Intentions, Plans, and Practical Reason*, Harvard University Press, Massachusetts, 1987.
- [2] M. Dastani & F. de Boer & F. Dignum and J.-J. Meyer. Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of The Second Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*, Melbourne, Australia, 2003.
- [3] M. Dastani & M. B. van Riemsdijk and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [4] D.C. Dennett, *The Intentional Stance*, MIT Press, Cambridge, Mass., 1987.
- [5] J.-J. Ch. Meyer, Reasoning about Emotional Agents, in *Proc.16th European Conf. on Artif. Intell. (ECAI 2004)* (R. Lpez de Mntaras & L. Saitta, eds.), IOS Press, 2004, pp. 129-133.
- [6] K. Oatley & J.M. Jenkins, *Understanding Emotions*, Blackwell Publishing, Malden/Oxford, 1996.
- [7] A. Ortony, & G. L. Clore and A. Collins, *The Cognitive Structure of Emotions*, Cambridge University Press, Cambridge, UK, 1988.
- [8] A.S. Rao and M.P. Georgeff, Modeling rational agents within a BDI-architecture, in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)* (J. Allen, R. Fikes & E. Sandewall, eds.), Morgan Kaufmann, 1991, pp. 473-484.
- [9] Y. Shoham, Agent-Oriented Programming, *Artificial Intelligence* 60(1), 1993, pp. 51-92.
- [10] A. Sloman, Motives, Mechanisms, and Emotions, in: *The Philosophy of Artificial Intelligence* (M. Boden, ed.), Oxford University Press, Oxford, 1990, pp. 231-247.
- [11] A. Sloman, What sort of architecture is required for a human-like agent?, Techn. Report CSRP-96-12, School of Computer Science and Cognitive Science Research Centre, Birmingham, 1996. Invited talk for Cognitive Modelling Workshop at AAA—I'96.
- [12] J. Tao & T. Tan and R.W. Picard, *Affective Computing and Intelligent Interaction*, LNCS 3784, Springer, Berlin, 2005.

Goal Types in Agent Programming

Mehdi Dastani and M. Birna van Riemsdijk and John-Jules Ch. Meyer¹

Abstract. This paper presents three types of declarative goals: perform goals, achieve goals, and maintain goals. The integration of these goal types in a simple but extendable logic-based agent-oriented programming language is discussed and motivated. The computational semantics for each goal type is presented by means of a transition system. It is shown that the presented semantics of the goal types ensure some desirable and expected properties.

1 Introduction

An essential characteristic of autonomous intelligent agents is their pro-active behaviour [7]. These agents are assumed to have goals for which they (pro-actively) decide actions to perform. Different logics have been proposed to characterize goals, to represent and reason about them, and to specify their relations to other agent concepts such as beliefs and actions [2, 3, 5]. These logics allow the specification of various agent types, i.e., agents that have a certain attitude towards their goals.

Motivated by these logics, various agent-oriented programming languages have been proposed [1]. In order to allow the implementation of various goal related agent types (i.e., agents that can have different attitudes towards their goals) existing programming languages provide constructs to implement various types of goals. For example, JACK provides programming constructs to implement, among others, test, achieve, insist, and maintain goals, Jason has achieve and test goals, and Jadex has achieve, query, perform and maintain goals [1]. The way in which goals are treated by these programming languages differs. In Jadex, goals are represented in XML in terms of a label/name and a number of other parameters. In Jason and JACK, goals are particular types of events. Further, neither JACK nor Jadex provides the formal semantics of their goal types. A detailed comparison between the goal types of the various languages is beyond the scope of this paper.

In this paper, we focus on three types of goals: perform goals, achieve goals, and maintain goals. These goals are represented as logical formulas having formal semantics. Perform goals can be used to generate plans without demanding that the plans *must* reach the states denoted by the goals. The attitude of an agent toward a perform goal is thus to generate relevant plans after which the goal will be dropped. For example, consider “FC and CC” as a perform goal of an agent where FC stands for having-fuel-in-car and CC for having-clean-car. The goal “FC and CC” can be used to generate, e.g., a plan to refuel at the gas station gs1 and a plan to clean the car in the car wash cw1. After generating the plans, the agent drops the goal entirely regardless of the plans’ effects. However, if the agent has only means to generate the plan to refuel, then it will only generate the refuel plan after which the entire goal is dropped. Dropping goals after generating relevant plans is a practical idea that has been im-

plemented in most agent-oriented programming languages, although sometimes under different names. In JACK and Jason this type of goal is known as achieve goal. As in Jadex, we associate a *redo* flag with a perform goal. If this flag is false, then the agent will behave as described above. Otherwise, if the flag is true, the agent will not drop its perform goal but apply relevant planning rules repeatedly and indefinitely. For example, a vacuum cleaner that has to clean a number of rooms repeatedly without the ability to check if a room is clean can be modelled as having a perform goal with a true redo flag.

The idea of an achieve goal is to reach the state denoted by it. Similar to perform goals, an agent with an achieve goal will apply relevant planning rules to generate and execute plans. If the achieve goal is not reached after the execution of these plans, it applies the planning rules again, hopefully generating different plans, since the circumstances might have changed. Once the achieve goal is reached, it will stop generating and executing plans for this goal. For example, consider an agent that has FC as an achieve goal and that can generate two plans, i.e., to refuel at gas stations gs1 and gs2. The agent can generate and execute the plan to refuel at gs1. If the plan is successful, the achieve goal will be dropped. Otherwise, it will generate and execute the second plan to refuel at gs2. If both plans do not achieve the goal, then it will apply the rules again. As suggested in [6] and implemented in Jadex, we assign a failure condition to each achieve goal to indicate when the agent should stop trying to achieve the goal and thus drop the goal. For example, no-fuel-at-gs1-and-gs2 can be the failure condition of the achieve goal FC. A similar type of goal is introduced in Jadex and in JACK under the name *insist* goal.

Finally, the idea of maintain goal is to ensure that a state holds and continues to hold. Plans should be generated and executed if the state denoted by the maintain goal is *threatened* not to hold, rather than waiting and taking action once the state does not hold. The condition under which the maintain goal is threatened not to hold will be called the maintain condition. The agent starts to generate and execute plans when the maintain condition becomes true. For example, consider an agent with FC as a maintain goal. This means that the agent wants to maintain having a fueled car. The maintain condition is the illuminated lamp warning of a shortage of fuel. The agent should generate and execute a refuel plan if the lamp is illuminated. If the maintain condition continues to hold after the execution of the plan, because for example the tank station had no fuel, the agent may try to generate and execute another plan, e.g., to go to another tank station to refuel. If all plans are generated and the maintain condition still holds, then there are two options. The agent can either stop generating and executing plans since it has tried all plans, or it can continue to apply the planning rules once again, hopefully generating new plans this time. In order to allow both options, we add a *retry* flag, like in Jadex [1], to the maintain goals. If the flag is true, the agent will try to re-apply planning rules, otherwise it does not.

¹ Utrecht University, The Netherlands, Email: {mehdi,birna,jj}@cs.uu.nl

2 Syntax

In order to implement different types of goals, we propose a simple but extendable logic-based agent-oriented programming language. We assume a propositional language L and the propositional entailment relation \models . A perform goal consists of a propositional formula and a *redo* flag that indicates whether the goal should be performed repeatedly. An achieve goal consists of two propositional formulas. The first formula denotes the state to be achieved and the second formula represents a failure condition. Finally, a maintain goal consists of two propositional formulas and a *retry* flag. The first formula denotes the state to be maintained and the second formula represents the maintain (trigger) condition. The retry flag indicates whether to repeat performing plans as long as the maintain trigger condition holds.

Definition 1 (*belief and goal languages*) The belief language L_σ , the perform goal language L_{γ_p} , the achieve goal language L_{γ_a} , and the maintain goal language L_{γ_m} are defined as follows:

- $L_\sigma = L$
- $L_{\gamma_p} = \{(\phi, f) \mid \phi \in L \& f \in \{\top, \perp\}\}$
- $L_{\gamma_a} = \{(\phi, fc) \mid \phi, fc \in L\}$
- $L_{\gamma_m} = \{(\phi, mc, f) \mid \phi, mc \in L \& f \in \{\top, \perp\}\}$

A plan is considered as a sequence of basic actions which update the beliefs of an agent when executed. The plan language can be extended with other actions such as test and communication, which can be composed by if-then-else and while constructs [4, 1].

Definition 2 (*plan*) Let *BasicAction* with typical element a be the set of basic actions. The set of plans *Plan* with typical element π is defined as: $\pi ::= a \mid \pi_1; \pi_2$. We use ϵ to denote the empty plan.

Planning rules are used for selecting an appropriate plan for a goal under a certain belief condition. A planning rule is of the form $\beta, \kappa \Rightarrow \pi$ and indicates to select plan π for the goal κ , if the agent believes β . In order to be able to check whether an agent has a certain belief or goal, we use propositional formulas from L to represent belief and goal query expressions.

Definition 3 (*plan selection rule*) The set of plan selection rules \mathcal{R}_{PG} is defined as: $\mathcal{R}_{PG} = \{\beta, \kappa \Rightarrow \pi \mid \beta, \kappa \in L, \pi \in \text{Plan}\}$.

In the following, we use $\mathbf{G}(r)$ and $\mathbf{B}(r)$ to indicate the goal condition κ and the belief condition β , respectively, that occur in the head of the planning rule $r = (\beta, \kappa \Rightarrow \pi)$.

Given these languages, an agent can be implemented by programming four sets of propositional formulas (representing the agent's beliefs, perform goals, achieve goals, and maintain goals), one set of planning rules, and an ordering on the set of planning rules to indicate the order in which the planning rules should be applied.

Definition 4 (*agent program*) An agent program is a tuple $(\sigma, \gamma_p, \gamma_a, \gamma_m, PG, <)$ where $\sigma \subseteq L_\sigma$, $\gamma_p \subseteq L_{\gamma_p}$, $\gamma_a \subseteq L_{\gamma_a}$, $\gamma_m \subseteq L_{\gamma_m}$, $PG \subseteq \mathcal{R}_{PG}$, and $<$ is a strict order on PG .

3 Semantics

The semantics of the programming language is defined by means of a transition system. A transition system for a programming language consists of a set of axioms and derivation rules for deriving transitions for this language. A transition is a transformation of one configuration into another and it corresponds to a single computation step. A configuration represents the state of an agent at each point during computation.

For the purpose of this paper, a configuration consists of a belief base σ representing the agent's beliefs, a goal base γ representing the agent's goals, a plan base Π containing the generated plans, a set of planning rules PG , and a set T to administrate the set of planning rules that have already been tried for application. These rules will not be applied anymore because either they have been applied already or the goals that occur in their heads are achieved. In the rest of this paper, we will call these rules *tried rules*.

Definition 5 (*agent configuration*) Let $\Sigma = \{\sigma \mid \sigma \subseteq L_\sigma, \sigma \neq \perp\}$, $\gamma_p \subseteq \{(\phi, f) \in L_{\gamma_p} \mid \phi \not\models \perp\}$, $\gamma_a \subseteq \{(\phi, fc) \in L_{\gamma_a} \mid \phi \not\models \perp\}$, and $\gamma_m \subseteq \{(\phi, mc, f) \in L_{\gamma_m} \mid \phi \not\models \perp\}$. An agent configuration is a tuple $(\sigma, \gamma, \Pi, PG, <, T)$ where $\sigma \in \Sigma$ is the belief base, $\gamma = (\gamma_p, \gamma_a, \gamma_m)$ is the goal base, $\Pi \subseteq (L \times L \times \text{Plan})$ is the plan base, $PG \subseteq \mathcal{R}_{PG}$ is a set of planning rules, $<$ is a strict order on PG , and $T \subseteq \{(\phi, R) \mid \phi \in L, R \subseteq \{r \mid r \in PG \& \phi \models \mathbf{G}(r)\}\}$ administrates the sets of tried planning rules.

The goal base γ in this definition is a 3-tuple consisting of three goal bases $\gamma_p, \gamma_a, \gamma_m$. These goal bases represent the agent's perform goals, achieve goals, and maintain goals, respectively. Moreover, the elements of the plan base are defined as 3-tuples consisting of a plan and two goal formulas that indicate the reasons for generating the plan. More specifically, a plan can be generated by applying a planning rule $\beta, \kappa \Rightarrow \pi$, if κ is a subgoal of an agent's goal ϕ , i.e., if $\phi \models \kappa$. The two formulas associated with a plan in a 3-tuple are the agent's goal ϕ and its subgoal κ based on which a planning rule is applied and the plan is generated. This means that we have $\forall (\phi, \kappa, \pi) \in \Pi : \phi \models \kappa$. Finally, the set T is added to the agent configuration in order to administrate the planning rules R that have been tried to be applied for each goal ϕ of the agent. This information will be used to avoid applying the same planning rules repeatedly.

The initial configuration of an agent can be built based on the agent program (definition 4) that specifies the initial beliefs, goals, planning rules, and the ordering on the rules. As noted, an agent does not have initial plans for simplicity reasons. This implies that the plan base is empty in the initial configuration. Also, because none of the planning rules have been tried to be applied before an agent starts to execute, the set of applied planning rules for each goal in the initial configuration is empty.

Definition 6 (*initial configuration*) Let $(\sigma, \gamma_p, \gamma_a, \gamma_m, PG, <)$ be an agent program. The initial configuration of the agent is $(\sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, PG, <, T)$, where $\Pi = \emptyset$, and $T = \{(\phi, \emptyset) \mid (\phi, f) \in \gamma_p \text{ or } (\phi, fc) \in \gamma_a \text{ or } (\phi, mc, f) \in \gamma_m\}$.

In the sequel, we omit the set of planning rules PG and the ordering $<$ in the agent configuration for reasons of presentation. Moreover, in the following we use \models_b which is defined as follows: $(\langle \sigma, \gamma, T \rangle \models_b \phi) \Leftrightarrow (\sigma \models \phi)$.

When executing an agent, planning rules will be selected and applied based on its goals, beliefs, and the ordering on the planning rules. The application of planning rules generates plans which can be selected and performed. Before introducing the transition rules to specify possible agent execution steps, we need to define what it means to perform a plan. For simplicity reasons, we assume here that the performance of a plan affects only the belief base. The effect of plans on the belief base is captured through an update operator τ , which takes the belief base and a basic action and generates the updated belief base. This update operator can be as simple as adding/deleting atoms to/from the belief base.

Definition 7 (*plan performance*) Let *BasicAction* be the set of basic actions, $\tau : \text{BasicAction} \times \Sigma \rightarrow \Sigma$ be a partial function implement-

ing a belief update operator, and $a; \pi$ be a plan consisting of action a followed by the plan π . The performance of a plan with respect to a belief base is defined by the function $Perform : \Sigma \times \text{Plan} \rightarrow \Sigma$ that updates the belief base consecutively by the sequence of actions involved in the plan, i.e., $Perform(\sigma, a) = \tau(\sigma, a)$ and $Perform(\sigma, a; \pi) = Perform(Perform(\sigma, a), \pi)$.

Note that $Perform$ is a partial function since the belief update operator is a partial function: we assume that the function $Perform$ evaluates to *undefined* if it is applied to an undefined input element. Given the function $Perform$, the following two transition rules (called EP_1 and EP_2) define plan execution.

$$\frac{(\phi, \kappa, \pi) \in \Pi \& \pi \neq \epsilon \& Perform(\sigma, \pi) = \sigma'}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma', \gamma, (\Pi \setminus \{(\phi, \kappa, \pi)\}) \cup \{(\phi, \kappa, \epsilon)\}, T \rangle}$$

$$\frac{(\phi, \kappa, \pi) \in \Pi \& \pi \neq \epsilon \& Perform(\sigma, \pi) = \text{undefined}}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \setminus \{(\phi, \kappa, \pi)\}, T \rangle}$$

The first transition rule (EP_1) captures the case where the plan π is successfully performed. The resulting configuration contains the empty plan and the belief base is updated appropriately. The second transition rule (EP_2) captures the case that the performance of the plan has failed. Note that in this case, the failed plan (ϕ, ψ, π) will be removed from the plan base.

In order to define the transitions for different types of goals, we first define the notion of *relevant* and *applicable* planning rules w.r.t. an agent's goal, and an auxiliary function called *next*. Intuitively, a planning rule is relevant for an agent's goal if it can contribute to the agent's goal, i.e., if the goal that occurs in the head of the planning rule is a subgoal of the agent's goal. A planning rule is applicable to an agent's goal if it is not applied yet, if it is relevant for that goal, and if the belief condition of the rule is entailed by the agent's configuration. Finally, the next function selects the *first applicable planning rule* for an agent's goal if there exists one, evaluates to *nil* if all relevant planning rules have been tried to be applied, and evaluates to *undefined* otherwise.

Definition 8 (*relevant, applicable, next planning rules*) Let $C = \langle \sigma, \gamma, \Pi, \text{PG}, <, T \rangle$ be an agent configuration and $\langle \phi, R \rangle \in T$. For the given configuration C and goal ϕ , the set of relevant and applicable planning rules, and the next function are defined as follows:

- $rel(\phi, C) = \{r \in \text{PG} \mid \phi \models \mathbf{G}(r)\}$
 - $app(\phi, C) = \{r \in rel(\phi, C) \mid r \notin R, \sigma \models \mathbf{B}(r)\}$
 - $next(\phi, C)$
- = r if $r \in app(\phi, C) \& \forall r' \in app(\phi, C) : r \neq r' \rightarrow r > r'$
= nil if $rel(\phi, C) = R$
= $undefined$ otherwise

In the following transition rules, we omit the reference to configuration C and assume that $rel(\pi)$, $app(\phi)$, and $next(\phi)$ are used in the context of the left-hand side configuration of the transition.

Proposition 1 Let $\langle \sigma, \gamma, \Pi, \text{PG}, <, T \rangle$ be an agent configuration where $\langle \phi, R \rangle \in T$. Then, $next(\phi) = \text{undefined}$ iff $rel(\phi) \neq \emptyset \wedge \forall r \in rel(\phi) : (\sigma \not\models \mathbf{B}(r) \wedge r \notin R)$.

In order to compare the behaviours of an agent for different goal types, we need to define an agent's execution. An execution of an agent is the sequence of configurations that can be generated by applying transition rules. Agent executions can be generated by a cyclic procedure where in each cycle each transition rule gets the opportunity to be applied, i.e., in each cycle each transition rule is selected once and, if possible, applied. Such a cyclic procedure is in fact a

round robin scheduling technique for selecting and applying transition rules. We call such a procedure for selecting and applying transition rules a *round robin procedure*.

Definition 9 (*agent execution*) An execution of an agent is a finite or infinite sequence $\langle C_1, C_2, \dots \rangle$, where $C_i \rightarrow C_{i+1}$ is a transition derived from the transition system for $i \in N$. A fair agent execution is an execution that is generated by a round robin procedure.

3.1 Perform Goals

The idea of perform goals is to allow the generation of plans by applying planning rules. A planning rule can be applied if the goal in its head is logically entailed by one of the agent's goals. We use thus logical formulas to allow reasoning about the goals to decide which planning rule to apply. The following transition rule (called P_1) selects and applies the first (w.r.t. the ordering $<$ on the rules) applicable planning rule. This transition rule can be applied if there are no plans generated for the subgoal that occurs in the head of the planning rule and if the subgoal is not achieved yet. The latter condition is for efficiency.

$$\frac{(\phi_p, f) \in \gamma_p \& next(\phi_p) = (\beta, \kappa \Rightarrow \pi) \& \exists \pi' \in \text{Plan} : (\phi_p, \kappa, \pi') \in \Pi \& \sigma \not\models \kappa}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \cup \{(\phi_p, \kappa, \pi)\}, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_p, R \rangle\}) \cup \{\langle \phi_p, R \cup \{\beta, \kappa \Rightarrow \pi\}\rangle\}$.

Below is the second transition rule for a perform goal (called P_2) to administrate planning rules as being tried if the subgoal that occurs in their heads has already been achieved. As noted this is for efficiency reasons.

$$\frac{(\phi_p, f) \in \gamma_p \& r \in \text{PG} \& \sigma \models \mathbf{G}(r)}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_p, R \rangle\}) \cup \{\langle \phi_p, R \cup \{r\}\rangle\}$.

The third transition rule for a perform goal (called P_3) removes all empty plans from the plan base. This transition rule makes it possible to apply transition rule P_1 once again to generate an alternative plan for the perform goal.

$$\frac{(\phi_p, f) \in \gamma_p \& (\phi_p, \psi, \epsilon) \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \setminus \{(\phi_p, \psi, \epsilon)\}, T \rangle}$$

The perform goal is enriched with a *redo* flag which can be true or false. If all relevant planning rules for a perform goal have been applied (i.e., if the next function evaluates to nil) and the *redo* flag is false, then the perform goal will be removed from the goal base and the set of tried planning rules for this perform goal is emptied (reset). However, when the *redo* flag is true, the perform goal remains in the goal base while the set of planning rules will be emptied. This ensures that the plans associated with the perform goal will be generated and performed again. These two cases are captured by the below two transition rules (called P_4 and P_5).

$$\frac{(\phi_p, \perp) \in \gamma_p \& next(\phi_p) = \text{nil} \& \exists \phi \in L \exists \pi \in \text{Plan} : (\phi_p, \phi, \pi) \in \Pi}{\langle \sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, T \rangle \rightarrow \langle \sigma, (\gamma'_p, \gamma_a, \gamma_m), \Pi, T \setminus \{\langle \phi_p, R \rangle\} \rangle}$$

where $\gamma'_p = \gamma_p \setminus \{(\phi_p, \perp)\}$.

$$\frac{(\phi_p, \top) \in \gamma_p \& next(\phi_p) = \text{nil} \& \exists \phi \in L \exists \pi \in \text{Plan} : (\phi_p, \phi, \pi) \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi, (T \setminus \{\langle \phi_p, R \rangle\}) \cup \{\langle \phi_p, \emptyset \rangle\} \rangle}$$

The above transition rules determine how an agent processes its perform goals. The following proposition shows that an agent with

a perform goal to which a false redo flag is assigned, can drop the goal if the relevant planning rules have tautologies as belief conditions. It shows that such an agent will drop the goal if its behaviour is generated by a fair execution procedure.

Proposition 2 Let $C = \langle \sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, T \rangle$ be an initial agent configuration, where $(\phi_p, \perp) \in \gamma_p$ and $\forall r \in \text{PG} : \mathbf{B}(r) = \top$. There exists a finite execution that removes (ϕ_p, \perp) from the goal base. All fair executions eventually remove (ϕ_p, \perp) from the goal base.

3.2 Achieve Goals

For an achieve goal plans should be generated to reach the state denoted by it. If the generated and performed plans do not achieve the desired state, then the achieve goal remains in the goal base. The first transition rule below (called A_1) is designed to apply planning rules in order to achieve the subgoals of the achieve goals. A planning rule can be applied if the goal in the head of the rule is not achieved yet, if there is no plan for the same subgoal in the plan base, and if the failure condition does not hold. The application of a planning rule will add the plan of the planning rule to the plan base and administrate it as a tried rule.

$$\frac{(\phi_a, fc) \in \gamma_a \& \text{next}(\phi_a) = (\beta, \kappa \Rightarrow \pi) \& \exists \pi' \in \text{Plan} : (\phi_a, \kappa, \pi') \notin \Pi \& \sigma \not\models \kappa \& \sigma \not\models fc}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \cup \{(\phi_a, \kappa, \pi)\}, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_a, R \rangle\}) \cup \{\langle \phi_a, R \cup \{(\beta, \kappa \Rightarrow \pi)\}\rangle\}$.

If the head of a planning rule is already achieved, then the rule will be considered as tried and will be administrated accordingly. This is realized by the next transition rule (called A_2).

$$\frac{(\phi_a, fc) \in \gamma_a \& r \in \text{PG} \& \sigma \models \mathbf{G}(r)}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_a, R \rangle\}) \cup \{\langle \phi_a, R \cup \{(\beta, \kappa \Rightarrow \pi)\}\rangle\}$.

The next transition rule (A_3) removes all empty plans, which are generated for the achieve goals, from the plan base. Note that this transition rule is similar to P_3 . We did not generalize this rule to be applicable to all goal types since the empty plans for the maintain goals should be removed differently (see next section).

$$\frac{(\phi_a, fc) \in \gamma_a \& (\phi_a, \psi, \epsilon) \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \setminus \{(\phi_a, \psi, \epsilon)\}, T \rangle}$$

An achieve goal can be dropped under two circumstances: either when the failure condition, which is assigned to it, becomes true, or when the state it denotes is reached. The transition rule A_4 below captures these two cases of dropping the achieve goal. In both cases, besides the removal of the achieve goal, the plans associated with it will be removed and the set of tried planning rules will be set to empty.

$$\frac{(\phi_a, fc) \in \gamma_a \& (\sigma \models \phi_a \text{ or } \sigma \models fc)}{\langle \sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, T \rangle \rightarrow \langle \sigma, (\gamma_p, \gamma'_a, \gamma_m), \Pi', T \setminus \{\langle \phi_a, R \rangle\} \rangle}$$

where $\gamma'_a = \gamma_a \setminus \{(\phi_a, fc)\}$ and $\Pi' = \Pi \setminus \{(\phi_a, \kappa, \pi) \mid \kappa \in L, \pi \in \text{Plan}\}$.

When all planning rules that could achieve a subgoal of an achieve goal have been applied, all generated plans have been performed, but the state denoted by the achieve goal is not reached, then the set of tried rules will be set to empty in order to enable a new round of rule applications and plan performance. This is realized by the following transition rule A_5 .

$$\frac{(\phi_a, fc) \in \gamma_a \& \text{next}(\phi_a) = \text{nil} \& \sigma \not\models \phi_a \& \sigma \not\models fc \& \exists \kappa \in L \exists \pi \in \text{Plan} : (\phi_a, \kappa, \pi) \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi, (T \setminus \{\langle \phi_a, R \rangle\}) \cup \{\langle \phi_a, \emptyset \rangle\} \rangle}$$

Given the transition rules A_1, \dots, A_5 for the achieve goals, we can now show that an achieve goal will not be removed from the goal base unless the state denoted by it is reached.

Proposition 3 Let $\langle C_1, C_2, \dots \rangle$ be an agent execution, where $C_i = \langle \sigma^i, (\gamma_p^i, \gamma_a^i, \gamma_m^i), \Pi^i, T^i \rangle$ and $(\phi, \perp) \in \gamma_a^1$. Then,
 $\forall k \in N : (\sigma^1 \not\models \phi \wedge \dots \wedge \sigma^k \not\models \phi) \rightarrow (\phi, \perp) \in \gamma_a^k$

The achievement of an achieve goal depends on many factors among which whether the planning rules are designed correctly and whether the plans of different rules interfere with each other. In order to define whether a planning rule is designed correctly, we need to specify the consequence of performing a plan in an agent configuration. The following function XP maps an agent configuration C to another agent configuration C' such that C' can be reached from configuration C by performing successfully the plan π (from the plan base of C), i.e., $C \rightarrow C'$ is a transition derived by applying the transition rule EP_1 .

Definition 10 (XP function) Let \mathcal{C} be the set of agent configurations. The eXecute Plan function $XP : \mathcal{C} \times (L \times L \times \text{Plan}) \rightarrow \mathcal{C}$ is defined as follows: $XP(C, (\phi, \kappa, \pi)) = C'$ iff $C \rightarrow C'$ is derivable by applying transition rule EP_1 through which (ϕ, κ, π) from the plan base of C is executed, i.e., where $C = \langle \sigma, \gamma, \Pi, T \rangle$, $C' = \langle \sigma', \gamma, \Pi', T \rangle$, $(\phi, \kappa, \pi) \in \Pi$, $\Pi' = (\Pi \setminus \{(\phi, \kappa, \pi)\}) \cup \{(\phi, \kappa, \epsilon)\}$, and $\text{Perform}(\sigma, \pi) = \sigma'$.

A correct planning rule is a rule for which the performance of its plan in an agent configuration achieves the goal occurring in its head.

Definition 11 (correct planning rules) Let XP be the eXecute Plan function. A planning rule $\beta, \kappa \Rightarrow \pi$ is correct iff for all configurations C and $\phi, \psi \in L$ it holds that: $XP(C, (\phi, \psi, \pi)) \models_b \kappa$

The applications of planning rules generate a set of plans that can be performed in an arbitrary order. If plans are not designed carefully, then their order of performance can have undesirable effects. In order to guarantee that a set of plans reach certain states independent of their order of performance, we employ a notion of non-interfering plans.

Definition 12 (non-interfering plans) Let $C = \langle \sigma, \gamma, \Pi, T \rangle$ be an agent configuration, $p_1 = (\phi_1, \psi_1, \pi_1)$, $p_2 = (\phi_2, \psi_2, \pi_2)$, and $p_1, p_2 \in \Pi$. The plans π_1 and π_2 are non-interfering iff
 $XP(XP(C, p_1), p_2) = XP(XP(C, p_2), p_1)$

If all planning rules are correct and they can always be applied (the belief condition of the rules are tautologies), then all fair executions reach eventually the state denoted by an achieve goal that is equivalent with the goals occurring in the head of the planning rules.

Proposition 4 Let $C_1 = \langle \sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, T \rangle$ be an initial configuration, $(\phi, fc) \in \gamma_a$, all planning rules in PG be correct, the plans in these rules be non-interfering, and there exists $\{(\top, \kappa_1 \Rightarrow \pi_1), \dots, (\top, \kappa_n \Rightarrow \pi_n)\} \subseteq \text{PG}$ such that $\{\kappa_1, \dots, \kappa_n\} \models \phi$ and $\phi \models \kappa_i$ for $1 \leq i \leq n$. Then, there exists a configuration C_n in every fair agent execution $\langle C_1, C_2, \dots \rangle$ such that $C_n \models_b \phi$.

The transition rules for the perform and achieve goals show close similarities. In order to examine and compare an agent's behaviours with respect to different goal types, we define goal-equivalent agent executions. The idea is that the consecutive agent configurations in these agent executions are identical in all components except their goal bases.

Definition 13 (*goal-equivalent executions*) Let $e_1 = \langle C_1, C_2, \dots \rangle$ and $e_2 = \langle C'_1, C'_2, \dots \rangle$ be agent executions, where for $i \in N$, $C_i = \langle \sigma_i, \gamma_i, \Pi_i, T_i \rangle$ and $C'_i = \langle \sigma'_i, \gamma'_i, \Pi'_i, T'_i \rangle$ are agent configurations. The executions e_1 and e_2 are goal-equivalent iff $\sigma_i = \sigma'_i$, $\Pi_i = \Pi'_i$, and $T_i = T'_i$ for $i \in N$.

Note that the goal-equivalent relation is an equivalence relation. We can now show that under certain conditions the achieve goal of an agent can be replaced with a perform goal. In particular, an agent with an achieve goal such that the state denoted by it cannot be reached and its failure condition never becomes true (e.g., the goal to have fuel in the car FC while there is no fuel anywhere, with the failure condition no-fuel-in-world assuming that the agent cannot come to conclude this belief), can be replaced by a perform goal with a true redo flag (e.g., the goal FC with a flag \top while there is no fuel anywhere).

Proposition 5 Let $C_1 = \langle \sigma, (\gamma_p, \gamma_a, \gamma_m), \Pi, T \rangle$ be an agent configuration. If there exists no execution $\langle C_1, C_2, \dots \rangle$ with $C_i \models_b \phi$ for some $i \in N$, then for every execution starting from $\langle \sigma, (\gamma_p, \gamma_a \cup \{(\phi, \perp)\}, \gamma_m), \Pi, T \rangle$ there exists a goal-equivalent execution starting at $\langle \sigma, (\gamma_p \cup \{(\phi, \top)\}, \gamma_a, \gamma_m), \Pi, T \rangle$.

3.3 Maintain Goals

The state denoted by a maintain goal should hold at any moment during the agent execution. In order to maintain the denoted state, plans should be generated and performed. The question is when exactly plans should be generated and performed. A maintain goal is enriched with a triggering condition which, if it becomes true, indicates that plans should be generated and performed. This triggering condition can be considered as an alarm to act in order to ensure the maintenance of the state denoted by the maintain goal. It should be noted that there might be no logical relation between the maintain goals and their triggering conditions. This relation depends usually on the application domain. However, we assume that in all agent configurations, if the triggering condition does not hold, then the maintain goal holds: $\forall \sigma \in \Sigma, \forall \phi_m, mc \in L : (\sigma \not\models mc) \rightarrow (\sigma \models \phi_m)$. We associate a *retry* flag to maintain goals. If this flag is set to false, all planning rules have been applied, the generated plans have been performed, and the maintain triggering condition still holds, then no new rounds of actions should be taken. This flag can thus be used to avoid new attempts to maintain a goal.

The first transition rule M_1 is designed to allow the application of planning rules when the maintain triggering condition becomes true. The applied rule should be such that there is no plan already in the plan base for the subgoal that occurs in the head of the selected rule.

$$\frac{(\phi_m, mc, f) \in \gamma_m \text{ & } next(\phi_m) = (\beta, \kappa \Rightarrow \pi) \text{ & } \exists \pi' \in \text{Plan} : (\phi_m, \kappa, \pi') \in \Pi \text{ & } \sigma \models mc}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \cup \{(\phi_m, \kappa, \pi)\}, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_m, R \rangle\}) \cup \{\langle \phi_m, R \cup \{(\beta, \kappa \Rightarrow \pi)\}\rangle\}$.

In order to maintain a goal, applicable planning rules should be applied independently of whether the goal in their heads, which are subgoals of the maintain goals, are achieved. But this means that all planning rules that have the same subgoal in their heads will be applied if they are applicable. This can, however, make the executions inefficient. For example, suppose that the maintain goal of an agent is to have fuel in the car FC and that the agent has several planning rules that indicate several plans to refuel. It is undesirable to allow the agent to apply all planning rules to generate all possible ways to refuel. Therefore, we administrate all planning rules with the same

goal in their heads as tried rules when one plan for this subgoal is successfully performed. This idea is captured by the transition rule M_2 .

$$\frac{(\phi_m, mc, f) \in \gamma_m \text{ & } \{(\phi_m, \kappa, \epsilon)\} \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi \setminus \{(\phi_m, \kappa, \epsilon)\}, T' \rangle}$$

where $T' = (T \setminus \{\langle \phi_m, R \rangle\}) \cup \{\langle \phi_m, R \cup \{r \in \text{PG} \mid \mathbf{G}(r) = \kappa\}\rangle\}$.

If there are no plans to perform, no rules to apply, and either the goal is maintained (maintain condition does not hold) or the retry flag is true, the administration of applied rules is reset. The reset makes it possible to re-apply rules and re-perform plans if needed. Note that if the maintain condition holds (actions should be taken to maintain the goal) and the retry flag is false, then the goal is not maintainable and the agent cannot do anything to maintain the goal.

$$\frac{(\phi_m, mc, f) \in \gamma_m \text{ & } next(\phi_m) = nil \text{ & } (f = \top \text{ or } \sigma \not\models mc) \text{ & } \exists \kappa \in L \exists \pi \in \text{Plan} : (\phi_m, \kappa, \pi) \in \Pi}{\langle \sigma, \gamma, \Pi, T \rangle \rightarrow \langle \sigma, \gamma, \Pi, (T \setminus \{\langle \phi_m, R \rangle\}) \cup \{\langle \phi_m, \emptyset \rangle\} \rangle}$$

Given these transition rules for the maintain goal, the perform goal (ϕ, \perp) and the maintain goal (ϕ, \top, \perp) are identical.

Proposition 6 Let σ be a belief base, for all $r \in \text{PG} : \sigma \not\models \mathbf{G}(r)$, all planning rules are correct, and plans are non-interfering. Then, for every execution starting from $C_1 = \langle \sigma, (\gamma_p, \gamma_a, \gamma_m \cup \{(\phi, \top, \perp)\}), \Pi, T \rangle$ there exists a goal-equivalent execution starting from $C'_1 = \langle \sigma, (\gamma_p \cup \{(\phi, \perp)\}, \gamma_a, \gamma_m), \Pi, T \rangle$.

4 Conclusion

In this paper, we have introduced three types of declarative goals for which we argued that they should be integrated in logic-based agent-oriented programming languages. We presented a simple agent-oriented programming language that allows the implementation of these goal types. The formal semantics of the programming language, and thus of the goal types, is given and it is shown that this semantics has desirable properties (because of the space limit we could not present the proofs). The presented programming language is simplified for the purpose of this paper. The language can be extended by using a (computational) subset of a first-order predicate language instead of using a propositional language. In this way, declarative goals become more expressive. Also, the plan language can be extended to allow more complex plans. We have started implementing an interpreter for an extended version of the presented programming language which we hope will be available soon. For this implementation, we have decided to generate a specific fair execution of agents by means of a deliberation cycle. In future work, we will investigate possible deliberation cycles and use temporal logics to prove properties of different deliberation cycles and to compare them.

REFERENCES

- [1] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, *Multi-Agent Programming: Languages, Platforms and Applications*, Springer, Berlin, 2005.
- [2] C. Boutilier, ‘Toward a logic for qualitative decision theory’, in *Proceedings of the KR’94*, pp. 75–86, (1994).
- [3] Philip R. Cohen and Hector J. Levesque, ‘Intention is choice with commitment’, *Artificial Intelligence*, **42**, (1990).
- [4] M. Dastani and L. van der Torre, ‘Programming BOID-Plan agents: deliberating about conflicts among defeasible mental attitudes and plans’, in *Proc. of AAMAS’04*, pp. 706–713, New York, USA, (2004).
- [5] Anand S. Rao and Michael P. Georgeff, ‘Modeling rational agents within a BDI-architecture’, in *Proc. of KR’91*, (1991).
- [6] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, ‘Declarative and procedural goals in intelligent agent systems’, in *Proc. of KR’02*, (2002).
- [7] Michael Wooldridge, *Introduction to Multiagent Systems*, John Wiley & Sons, Inc., 2002.

Alternating-Offers Bargaining under One-Sided Uncertainty on Deadlines

Francesco Di Giunta and Nicola Gatti¹

Abstract. Alternating-offers is the most prominent negotiation protocol for automatic bilateral bargaining. Nevertheless, in most settings it is still not known how two fully rational agents should behave in the protocol. In this paper we study the finite-horizon alternating-offers protocol under one-sided uncertain deadlines. We make a novel use of backward induction in studying bargaining with uncertainty; we employ a “natural” system of beliefs and find, when it exists, the pertinent pure strategy sequential equilibrium. We further show, as an intrinsic limitation of the protocol, that for some parameter values there is no pure strategy sequential equilibrium, whatever system of beliefs is employed.

1 Introduction

Automated negotiation is a prominent research area of distributed artificial intelligence which studies the processes whereby rational software agents try to solve disputes and reach mutually beneficial agreements [7]. It is well known that the automation of the negotiating agents leads to more effective negotiations since agents are more efficient than humans in finding optimal agreements [14].

Among the negotiation settings for commercial transactions, a very common one is *bargaining*: a buyer and a seller try to agree on the choice of the value of some variables, named *issues*, that define the transaction they are carrying out (e.g. the price and quantity of a traded good). The formalized study of bargaining is commonly carried out with game-theoretical tools [9] in which one distinguishes the negotiation *protocol* and the negotiation *strategies*: the protocol sets the negotiation rules, specifying which actions are allowed and when [10]; the strategy defines an agent’s possible specific behavior in the negotiation. Given a protocol, rational agents should employ strategies that maximize their payoffs, and the classic game-theoretic approach prescribes that agents employ equilibrium strategies, where the notion of equilibrium is Nash equilibrium or extensions [5].

The best known protocol for bilateral bargaining, the *alternating-offers* protocol, pioneered by Ståhl [16], has reached an outstanding place in literature thanks to Rubinstein [11], and comes in many variations. Basically, an agent starts by offering values for the issues under dispute to her opponent, who can accept or make a counteroffer or exit the negotiation. If a counteroffer is made, the process is repeated until one of the agents accepts or exits the negotiation. There can be one or more bargained issues, leading respectively to *one-issue* and *multi-issue* bargaining. The issues are usually continuous, i.e. they are real-valued variables. The agents have reservation values, temporal utility discount factors, and negotiation deadlines. If the agents have finite negotiation deadlines, the protocol is a *finite-horizon* one; otherwise, it is *infinite-horizon*.

Although much economics and computer science literature deals with the alternating-offers protocol (see e.g. [7]), several major problems are still open, the main ones concerning *multi-issue* and *incomplete information* bargaining. Easy and general solutions are currently available when the bargaining is on one issue and every pertinent information is common knowledge between the two agents, but both assumptions are very restrictive. The problem of efficient multi-issue bargaining has been addressed in [2]. The problem of incomplete knowledge is harder. Rubinstein [12] has provided the first results about uncertainty over two possible time discount factors of one of the two agents. Several other authors have followed his research line, providing results on very narrow lacks of information completeness. We refer to [1] for a survey.

Given the difficulties in finding classic solutions to alternating-offers problems with incomplete information, several authors have proposed non-standard approaches. The best known proposal is Fátima, Wooldridge, and Jennings’s framework in [4]. Their approach is based on the *negotiation decision functions* paradigm [3], where the agents are supposed to constraint themselves to employ one of some predefined bidding tactics in their bargaining.

The aim of our paper is to analyze and solve the finite-horizon alternating-offers protocol under incomplete information on the bargaining deadline of one of the two agents (*one-sided* incomplete information). We adopt a classical approach, where the agents are not *a priori* assumed self-restricted to any class of strategies, in the belief that this approach is more coherent with the multiagent system paradigm (see, e.g., the discussion in [15]). Furthermore, in accordance with virtually all the literature on the topic, we employ pure strategies rather than mixed ones.

Our original contributions are: (1) the solution of the single-issue finite-horizon alternating-offers bargaining under one-sided uncertainty on deadlines, for a wide range of bargaining parameters; (2) the proof of non-existence of a general solution in pure strategies.

The paper is structured as follows: in the next section we review the finite-horizon alternating-offers protocol with complete information and its well known solution, in order to introduce basic concepts and state notation; in Section 3 we analyze the protocol under one-sided uncertain deadlines; in Section 4 we show that the general problem has no solution in pure strategies; Section 5 summarizes our conclusions.

2 Complete information alternating-offers

The protocol we study is a discrete time finite-horizon alternating-offers bargaining protocol on one continuous issue (say, a price).²

¹ Politecnico di Milano, Italy, email: {digiunta, ngatti}@elet.polimi.it

² The multi-issue problem [2] is orthogonal to the incomplete information problem. So, for the sake of simplicity, we consider one-issue bargaining.

Formally, the buyer b and the seller s can act at times $t \in \mathbb{N}$. The *player function* $\iota : \mathbb{N} \rightarrow \{b, s\}$ returns the agent that acts at time t and is such that $\iota(t) \neq \iota(t+1)$. Possible actions $\sigma_{\iota(t)}^t$ of agent $\iota(t)$ at time $t > 0$ are:

- (1) *offer*(\bar{x}), where $\bar{x} \in \mathbb{R}$,
- (2) *exit*,
- (3) *accept*.

At $t = 0$ the only allowed actions are (1) and (2). If $\sigma_{\iota(t)}^t = \text{accept}$ the bargaining stops and the *outcome* is (\bar{x}, t) , where \bar{x} is the tuple such that $\sigma_{\iota(t-1)}^{t-1} = \text{offer}(\bar{x})$. If $\sigma_{\iota(t)}^t = \text{exit}$ the bargaining stops and the outcome is *NoAgreement*. Otherwise the bargaining continues to the next time point.

Each agent i has an utility function $U_i : (\mathbb{R} \times \mathbb{N}) \cup \{\text{NoAgreement}\} \rightarrow \mathbb{R}$, that represents her gain on the possible bargaining outcomes. Each utility function U_i depends on three parameters of agent i :

- the *reservation price* $RP_i \in \mathbb{R}^+$,
- the *temporal discount factor* $\delta_i \in (0, 1]$,
- the *deadline* $T_i \in \mathbb{N}$, $T_i > 0$.

Exactly, if the outcome of the bargaining is an agreement (x, t) , then the utility functions U_b and U_s are respectively:

$$U_b(x, t) = \begin{cases} (RP_b - x)\delta_b^t & \text{if } t \leq T_b \\ -1 & \text{otherwise} \end{cases}, \quad U_s(x, t) = \begin{cases} (x - RP_s)\delta_s^t & \text{if } t \leq T_s \\ -1 & \text{otherwise} \end{cases}.$$

If the outcome is *NoAgreement*, then $U_b(\text{NoAgreement}) = U_s(\text{NoAgreement}) = 0$.³

Some standard hypothesis are assumed. *Feasibility*: $RP_b \geq RP_s$. *Rationality*: it is common knowledge that each agent will act to maximize her utility. *Benevolence*: it is common knowledge that if an agent has to choose between two outcomes which are indifferent for her but not for her opponent, she will choose the one that is better for her opponent.

The bargaining strategies that the agents should select also depend on the knowledge that the agents have of the protocol and of each other's utility function. In *complete information* bargaining it is assumed that the protocol and the utility functions (including the values of RP_i , δ_i and T_i) are common knowledge between the two agents.

The appropriate notion of solution for a complete information extensive form game like the one we are dealing with is *subgame perfect equilibrium* [6]. Subgame perfect equilibrium strategies can be easily found by *backward induction*, as we discuss in the following.

No agent is willing to bargain after her deadline, when any agreement would have negative utility. Therefore, at time $\bar{T} = \min\{T_b, T_s\}$ the acting agent – let's say s – would accept any offer with non-negative utility. Therefore, at time $\bar{T} - 1$ her opponent b could safely offer RP_s (which would be accepted) or could accept any possible previous offer x which is not worse than offering RP_s (i.e., $U_b(x, \bar{T} - 1) \geq U_b(RP_s, \bar{T})$). Therefore, at time $\bar{T} - 2$ agent s could safely offer the maximum x such that $U_b(x, \bar{T} - 1) \geq U_b(RP_s, \bar{T})$, i.e. x such that $U_b(x, \bar{T} - 1) = U_b(RP_s, \bar{T})$, or accept any possible previous offer which is not worse than offering x . This reasoning can be inductively carried on until the beginning of the game, finding an offer that agent $\iota(0)$ would do and her opponent would accept.

At each time point t , from \bar{T} back, it is therefore possible to know which offer would be made by agent $\iota(t)$ if she would make an offer:

³ Notice that employment of $U_i = -1$ after agent i 's deadline allows a rational behaviour of the agent after her deadline: an agent prefers to exit the negotiation than to reach any agreement.

we denote this series of offers by $x^*(t)$. In order to provide a recursive formula for $x^*(t)$, we introduce the notion of backward propagation: given value x and agent i , we call *backward propagation* of value x for agent i the value y such that $U_i(y, t-1) = U_i(x, t)$; we will employ the arrow notation $x \leftarrow_i$ for backward propagations.⁴ The calculation of $x^*(t)$ can now be stated like this:

$$x^*(t-1) = \begin{cases} RP_{\iota(t)} & \text{if } t = \bar{T} \\ (x^*(t))_{\iota(t)} & \text{if } t < \bar{T} \end{cases},$$

and is computationally linear with \bar{T} . A backward induction and the relevant $x^*(t)$ values can be plotted on the space (x, t) as in Figure 1.

The backward induction reasoning so far sketched proves the following result [9]:

Proposition 2.1 *Finite-horizon alternating-offers over single-issue bargaining with complete information has one and only one subgame perfect equilibrium. The equilibrium strategies for $t < \bar{T}$ are*

$$\sigma_{\iota(t)}^t = \begin{cases} \text{accept} & \text{if } \begin{cases} t \geq 0 \\ \sigma_{\iota(t-1)}^{t-1} = \text{offer}(x) \text{ with } U_{\iota(t)}(x, t) \geq U_{\iota(t)}(x^*(t), t+1) \end{cases} \\ \text{offer}(x^*(t)) & \text{otherwise} \end{cases}$$

and when $t = \bar{T}$

$$\sigma_{\iota(\bar{T})}^{\bar{T}} = \begin{cases} \text{accept} & \text{if } \sigma_{\iota(\bar{T}-1)}^{\bar{T}-1} = \text{offer}(x) \text{ with } U_{\iota(\bar{T})}(x, \bar{T}) \geq 0 \\ \text{exit} & \text{otherwise} \end{cases}.$$

The agreement is therefore achieved at time $t = 1$ on the price $x^*(0)$.

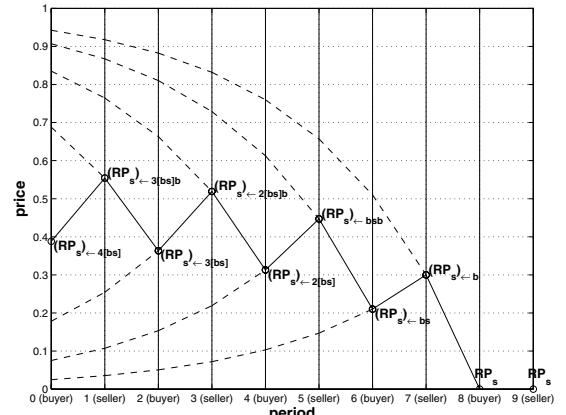


Figure 1. Backward induction with $RP_b = 1$, $RP_s = 0$, $\delta_b = 0.7$, $\delta_s = 0.7$, $T_b = 9$, $T_s = 10$, $\iota(0) = b$

3 Alternating-Offers with Uncertain Deadlines

Let us consider one-issue alternating-offers bargaining when one of the two agents does not exactly know her opponent's deadline. We will assume that the uncertain deadline is the buyer's.

As is customary in game theory in order to avoid underdetermined problems, we assume that b 's possible deadlines are distributed according to a probability distribution on \mathbb{R}^+ which is common knowledge between the agents. We further assume that the support of b 's deadline distribution is bounded; and since the agents

⁴ If a value x is backward propagated n times along the level curves of agent i , we write $x \leftarrow_{i[n]}$. If a value is backward propagated along the level curves of more than one agent, we list them left to right in the subscript; for instance, $x \leftarrow_{i3[j]}$ is value x backward propagated along the level curves of agent i and subsequently three times along the curves of agent j .

can act only at times $t \in \mathbb{N}$, we can assume, without loss of generality, that b 's deadline has a finite distribution on \mathbb{N} . We denote by $\mathcal{T}_b = \{T_{b_1}, \dots, T_{b_m}\}$ the set of possible deadlines of b , by $\mathcal{P}_b^0 = \{\alpha_{b_1}^0, \dots, \alpha_{b_m}^0\}$ the pertinent probability distribution, and by BT_b^0 the couple $BT_b^0 = \langle \mathcal{T}_b, \mathcal{P}_b^0 \rangle$. Agent b 's real deadline is known only to b itself: it is her *private information*.

In games with uncertainty, rational agents try to maximize their expected utilities relying on their beliefs about the opponent's private information and such beliefs are updated during the game, depending on which actions have been actually undertaken. The set of beliefs held by each agent over the other's private information after every possible sequence of actions in the game is called a *system of beliefs* and is usually denoted by μ . These beliefs are probabilistic and their values at time $t = 0$ is a given problem data. How beliefs evolve during the game, instead, is part of the solution which should be found for the game. A solution to an incomplete information bargaining is therefore a suitable couple $a = \langle \mu, \sigma \rangle$ called *assessment*.

An assessment $a = \langle \mu, \sigma \rangle$ must be such that the strategies in σ are mutual best responses given the probabilistic beliefs in μ (*rationality*); and the beliefs in μ must reasonably depend on the actions prescribed by σ (*consistency*). Different notions of solution differ on how they specify these two requirements.

We will employ the most common notion of solution for incomplete information bargaining, namely the the *sequential equilibrium* of Kreps and Wilson [8]. For a sequential equilibrium $a^* = \langle \mu^*, \sigma^* \rangle$, with $\sigma^* = \langle \sigma_b^*, \sigma_s^* \rangle$, the rationality requirement is specified as *sequential rationality*. Informally, after every possible sequence of actions S , on or off the equilibrium path, the strategy σ_s^* must maximize s 's expected utility given s 's beliefs prescribed by μ for S , and given that b will there on act according to σ_b^* ; and *vice versa*. The notion of consistency is defined as follows: assessment a is *consistent in the sense of Kreps and Wilson* (or simply *consistent*) if there exists a sequence a_n of assessments, each with completely mixed strategies and such that the beliefs are updated according to Bayes rule, that converges to a .

The method we will employ to find sequential equilibria is: (1) *a priori* fix a reasonable system of beliefs $\bar{\mu}$; (2) use backward induction to find, if they exist, the strategies σ of sequentially rational assessments a with beliefs $\bar{\mu}$; (3) *a posteriori* prove the consistency of the assessment.

We will use a system of beliefs $\bar{\mu}$ which is easy and natural: after any sequence of actions, agent s just excludes those deadlines T_{b_h} , among the initially possible ones, that have already expired and normalizes the probabilities of the future ones. Thus, in particular, the beliefs about the deadlines after a sequence of actions S depend only on the length of S . For it will be useful later, we introduce the *deadline function* $d(t)$, whose value is the probability, given at time t according to $\bar{\mu}$, that time t itself is a deadline for agent b .

Given the system of beliefs $\bar{\mu}$, we can find the sequentially rational strategies by backward induction. But the use of backward induction in this context is more involved than in the complete information bargaining and requires some explanations.

With complete information the backward induction can start at the earlier of the two deadlines; but with our incomplete information framework, the earlier deadline is not *a priori* known. Nevertheless, the backward induction can start at $\bar{T} = \min\{\max_h\{T_{b_h}\}, T_s\}$, because it is *a priori* known that after time \bar{T} agent b will exit the negotiation; if the bargaining process would reach time \bar{T} , then agent $\iota(\bar{T})$ would accept any nonnegative utility offer; therefore, at time $\bar{T} - 1$ agent $\iota(\bar{T} - 1)$, if she would make an offer, would offer $\iota(\bar{T})$'s reservation price.

It is therefore possible, like in the complete information setting of Section 2, to backward infer the offers $x^*(t)$ that the agents would do if they would choose to make an offer. But there are some complications in the construction of the optimal offers $x^*(t)$.

In the complete information case the values $x^*(t)$ are (i) the optimal offers, (ii) the values to be backward propagated, and (iii) the backward propagated values. In incomplete information bargaining, instead, this three series of values are distinct in general. Let us see these topics in more detail.

First, in complete information bargaining the optimal offer $x^*(t)$ is simply the backward propagation of $x^*(t + 1)$; with incomplete information this is not generally true: the backward propagated value is only the best among the surely accepted offers. But if at a time t , such that $\iota(t) = s$, there is a high probability that time $t + 1$ is a deadline of b , then agent s could prefer to offer RP_b rather than offer the best among the surely accepted offers, although offer RP_b could be rejected (if $t + 1$ is not b 's real deadline).

Second, as optimal offers of agent $s = \iota(t + 1)$ could be rejected, the value that should be backward propagated from time $t + 1$ to time t , in order to obtain the best one among the offers of b at time t that would be surely accepted by s , is not $x^*(t + 1)$ in general. The right value to be backward propagated is the value that we call *equivalent value* and is defined as follows: given an offer x of s , the equivalent value of x , denoted by $e(t)$, is the value such that $EU_s(e(t), 0) = EU_s(x)$.⁵ If offered value $(e^*(t + 1))_{\leftarrow s}$ at time t , agent s would accept it instead of offering $x^*(t + 1)$; if offered a worse value, she would refuse it and counteroffer $x^*(t + 1)$.

Summarily, backward propagation should be applied in general to equivalent values e^* rather than to best offers x^* , and best offers x^* of agent s are not always the backward propagated values. Formulas to find equivalent values and best offers are easy to find. If $\iota(t) = b$ then $e^*(t) = x^*(t)$ and $x^*(t) = (e^*(t + 1))_{\leftarrow s}$. Conversely, if $\iota(t) = s$ then $e^*(t) = RP_s + \delta^{t-1} \cdot EU_s(x^*(t))$ and $x^*(t) = \arg \max_{x=RP_b, (e^*(t+1))_{\leftarrow b}} \{EU_s(x)\}$ where:

$$\begin{aligned} EU_s(RP_b) &= (1 - d(t)) \cdot \left(d(t + 1) \cdot U_s(RP_b, t + 1) + (1 - d(t + 1)) \cdot \right. \\ &\quad \left. U_s(x^*(t + 1), t + 2) \right) \\ EU_s((e^*(t + 1))_{\leftarrow b}) &= (1 - d(t)) \cdot U_b((e^*(t + 1))_{\leftarrow b}, t + 1) \end{aligned}$$

Given $x^*(\cdot)$, the optimal strategy σ_i^* for agent i at time t is: if t is i 's deadline and at $t - 1$ she received an offer that gives her negative utility, then exit; if t is i 's deadline and at $t - 1$ she received an offer that gives her nonnegative utility, then accept; if t is not i 's deadline and at time $t - 1$ she received an offer not worse for her than $(x^*(t))_{\leftarrow i}$, then accept; otherwise offer $x^*(t)$.

The above backward construction provides the sequentially rational strategies with the system of beliefs $\bar{\mu}$ we have chosen, when they exist. But actually, they could not exist with some parameter setting. This can happen because one agent (precisely, the buyer) might deviate from strategy σ^* to offer something unacceptable for the opponent in order to be refused and to later be in a much stronger position and gain more. An example is given in Fig. 2. Here the backward induction construction imposes that at time 6 an offering buyer should make offer $e_{\leftarrow s}^7$; but it is easily seen that it is more convenient for her to offer something unacceptable (e.g., RP_s), then receive counteroffer RP_b , and finally re-counteroffer RP_s at time 8 (that would be surely accepted).

This anomaly arises because different types of buyers might have different optimal offers, thus revealing their private information and thus not allowing the seller to adopt the simple system of beliefs $\bar{\mu}$.

⁵ Clearly, the equivalent value of a surely accepted offer of s is the offer itself.

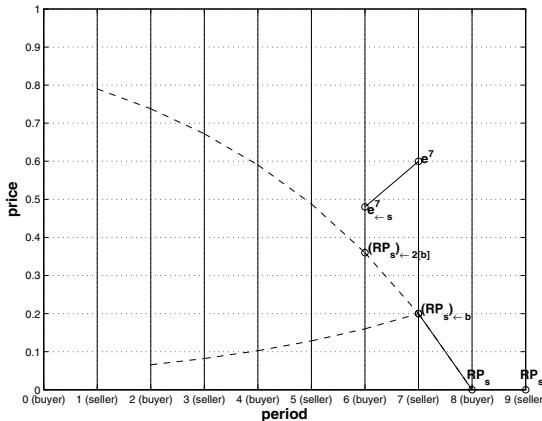


Figure 2. Non-existence of equilibrium with $RP_b = 1$, $RP_s = 0$, $\delta_b = 0.8$, $\delta_s = 0.8$, $\langle T_b = \{5, 8, 9\}, P_b^0 = \{0.5, 0.3, 0.2\}\rangle$, $T_s = 10$, $\iota(0) = b: x_{\leftarrow 2[b]} < e_{\leftarrow s}^7$

This happens when an “early deadline buyer” would offer the backward propagation of the seller’s equivalent value at the next time point, while a “late deadline buyer” would let time pass by to reach the part of the bargaining process where backward induction prescribes her utility to be higher. Precisely, this happens when t is a possible deadline of b , the optimal strategy of s at time $t - 1$ is to offer RP_b , and $U_b(x^*(t - 2), t - 2) < U_b(x^*(t), t)$. In other words it is not sequentially rational for b to offer $x^*(t - 2)$, but it is more convenient to deviate from the $x^*(t - 2)$ to offer $x^*(t)$ at t . This is the condition for the sequential rationality of strategies σ_i^* .

In Fig. 3 we depict an example of bargaining under one-sided uncertain deadlines that satisfies the above condition, so that the devised strategies are sequentially rational.

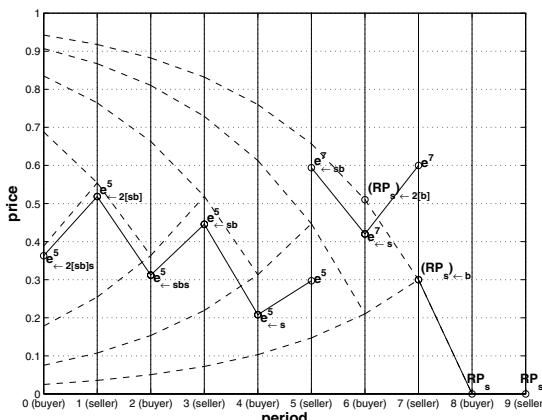


Figure 3. Backward induction with $RP_b = 1$, $RP_s = 0$, $\delta_b = 0.7$, $\delta_s = 0.7$, $\langle T_b = \{5, 8, 9\}, P_b^0 = \{0.5, 0.3, 0.2\}\rangle$, $T_s = 10$, $\iota(0) = b$: the existence is given by $e_{\leftarrow s}^7 < x_{\leftarrow 2[b]}$

We can now state the following:

Theorem 3.1 If for all t such that $\iota(t) = b$ holds $U_b(x^*(t - 2), t - 2) \geq U_b(x^*(t), t)$, then the assessment $a = \langle \bar{\mu}, \sigma^* \rangle$ above described

is a sequential equilibrium.

Proof sketch. The sequential rationality is easily seen from the above discussion. Consistency can be proved by the assessment sequence $a_n = \langle \mu_n, \sigma_n \rangle$ where:

- σ_n is the totally mixed strategy profile such that before the real deadline of an agent there is probability $1 - \frac{1}{n}$ of performing the action prescribed by σ and the remaining probability $\frac{1}{n}$ is uniformly distributed among the other allowed actions; while at the deadline or after it there is probability $1 - \frac{1}{n^2}$ of performing the action prescribed by σ and the remaining probability $\frac{1}{n^2}$ is uniformly distributed among the other allowed actions.
- μ_n is the system of beliefs obtained applying Bayes rule starting from the same *a priori* probability distribution P_b^0 as in $\bar{\mu}$.

Each assessment a_n is “Bayes consistent” by construction. The convergence of σ_n to σ^* is trivial. As to μ_n , given any arbitrary legal sequence S of actions (such that the bargaining does not end at the end of S and such that s is the agent acting after S), call $P_n^S = \langle \alpha_{n,b_1}^S, \alpha_{n,b_2}^S, \dots, \alpha_{n,b_m}^S \rangle$ the probabilities that agent s assigns to b ’s deadlines after sequence S according to μ_n . Sequence S might contain actions that could be interpreted as being in accordance to the strategies σ^* (i.e. actions that are the actions prescribed by strategies σ^* for some deadline T_{b_i}); be τ the time of the latest such action in S (if there are no such actions, set $\tau = -1$ by convention). Be $t = |S|$. Some calculation shows that, if $t \leq \bar{T}$, then

$$\alpha_{n,b_i}^S = \begin{cases} 0 & \text{if } T_{b_i} \leq \tau \\ \frac{\frac{1}{n^{t-T_{b_i}}} \cdot \alpha_{b_i}^0}{\sum_{T_{b_h} \geq t} \alpha_{b_h}^0 + \sum_{\tau < T_{b_h} \leq t} \frac{1}{n^{t-T_{b_h}}} \cdot \alpha_{b_h}^0} & \text{if } \tau < T_{b_i} \leq t \\ \frac{\alpha_{b_i}^0}{\sum_{T_{b_h} > t} \alpha_{b_h}^0 + \sum_{\tau < T_{b_h} < t} \frac{1}{n^{t-T_{b_h}}} \cdot \alpha_{b_h}^0} & \text{if } t < T_{b_i} \end{cases}$$

Therefore

$$\lim_{n \rightarrow +\infty} \alpha_{n,b_i}^S = \begin{cases} 0 & \text{if } T_{b_i} \leq t \\ \frac{\alpha_{b_i}^0}{\sum_{T_{b_h} \geq t} \alpha_{b_h}^0} & \text{if } t < T_{b_i} \end{cases}$$

Therefore P_n^S converges to the beliefs prescribed by $\bar{\mu}$ in S . Thus μ_n converges to $\bar{\mu}$. \square

4 Unsolvable Bargaining

The non-existence problem is an intrinsic limitation of the protocol. In fact, alternating-offers protocol under uncertain deadlines does not always admit pure strategy sequential equilibria. The sequential equilibrium theory assures the existence of at least one sequential equilibrium in behavioural strategies [8], but the use of behavioural strategy equilibria is not considered fully satisfying [13].

In this section we produce an example of bargaining that does not admit any pure strategy sequential equilibrium. The example does not even admit a *weak sequential equilibrium* [5] (a weaker notion of sequential equilibrium). Basically, an assessment is a weak sequential equilibrium if it is sequentially rational and *weakly consistent* (i.e., consistent on the equilibrium path).

Our example is depicted in Fig. 4. The b ’s deadline can be either at time 2 (buyer b_{early}) or at time 10 (buyer b_{late}). Shown in the figure is the backward induction construction that would have been produced by our method, but this does not lead to a solution, because if agent b ’s real deadline would be 10, she would not offer $e_{\leftarrow s}^1$ at

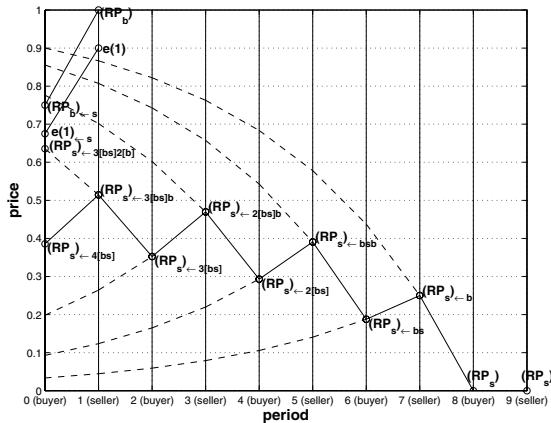


Figure 4. Non-existence of equilibrium with $RP_b = 1$, $RP_s = 0$, $\delta_b = 0.75$, $\delta_s = 0.75$, $\langle T_b = \{2, 10\}, P_b^0 = \{0.9, 0.1\} \rangle$, $T_s = 9$, $\iota(0) = b$

time $t = 0$ but would find more convenient to offer something unacceptable for s in order to offer $x_{\leftarrow 3[bs]}$ at time $t = 2$. In what follows we prove by contradiction that no pure strategy assessment works.

Theorem 4.1 *Alternating-offers bargaining with uncertain deadlines does not always admit a (weak) sequential equilibrium in pure strategies.*

Proof. Consider the bargaining of Fig. 4. By contradiction, be $a^* = \langle \sigma^*, \mu^* \rangle$ a pure strategy weak sequential equilibrium. At any time $t > 2$ the only possible system of beliefs for agent s at the equilibrium prescribes that the deadline of b is 10; therefore, for $t \geq 2$ the equilibrium strategies must be those prescribed by the backward induction construction. At time $t = 1$, instead, the beliefs of s at the equilibrium depend on which are the equilibrium actions of agent b at time $t = 0$. Two cases are possible (i) $\sigma_{b_{early}}^*(0) = \sigma_{b_{late}}^*(0)$ or (ii) $\sigma_{b_{early}}^*(0) \neq \sigma_{b_{late}}^*(0)$. We treat the two cases separately.

(i) If $\sigma_{b_{early}}^*(0) = \sigma_{b_{late}}^*(0)$, then by Bayes rule the beliefs of s at $t = 1$ on the equilibrium path are the initial ones (i.e., $P_b = \{0.9, 0.1\}$). Therefore, if at $t = 0$ agent b acts at the equilibrium, then at $t = 1$ the optimal strategy of agent s is to accept any offer greater than or equal to $e_{\leftarrow s}^1$ and otherwise offer RP_b . Thus, at time $t = 0$ the optimal strategy for b_{early} is to offer $e_{\leftarrow s}^1$; instead, at $t = 0$ the optimal strategy for b_{late} is to offer anything lesser than $e_{\leftarrow s}^1$ in order to be rejected and then, at $t = 2$ offer $x_{\leftarrow 3[bs]}$ that will be accepted. So $\sigma_{b_{early}}^*(0) \neq \sigma_{b_{late}}^*(0)$.

(ii) If $\sigma_{b_{early}}^*(0) \neq \sigma_{b_{late}}^*(0)$ then by Bayes rule the beliefs of s at $t = 1$ on the equilibrium path are: if action $\sigma_{b_{early}}^*(0)$ is observed, then $P_b = \{1, 0\}$; if action $\sigma_{b_{late}}^*(0)$ is observed, then $P_b = \{0, 1\}$. Therefore, if at $t = 0$ agent b acts at the equilibrium, then at $t = 1$ the optimal strategy of agent s is:

- if $\sigma_{b_{early}}^*(0)$ is observed, then accept any offer greater than or equal to $(RP_b)_{\leftarrow s}$ and otherwise offer RP_b ;
- if $\sigma_{b_{late}}^*(0)$ is observed, then accept any offer greater than or equal to $x_{\leftarrow 4[bs]}$ and otherwise offer $x_{\leftarrow 3[bs]b}$.

Thus, at time $t = 0$ the optimal strategy for both b_{early} and b_{late} is to offer $x_{\leftarrow 4[bs]}$; so $\sigma_{b_{early}}^*(0) = \sigma_{b_{late}}^*(0)$.

In conclusion it is neither $\sigma_{b_{early}}^*(0) = \sigma_{b_{late}}^*(0)$ nor $\sigma_{b_{early}}^*(0) \neq \sigma_{b_{late}}^*(0)$. Hence the pure strategy assessment $a^* = \langle \sigma^*, \mu^* \rangle$ is not a weak sequential equilibrium. \square

5 Conclusions

In this work we studied the effect of one-sided uncertain deadlines in the alternating-offers bargaining protocol with agents whose strategies are not self-constrained to any particular set of tactics. Although our study considers only one-sided uncertainty, the obtained results are particularly significant: (1) the equilibrium strategies can be found, when they exist, employing a simple backward procedure; (2) alternating-offers protocol does not always admit equilibrium in pure strategies.

The first result states that, for a wide range of bargaining parameters, the determination of equilibrium strategies of fully rational agents with one-sided uncertain deadlines can be simply tackled and the equilibrium prescribes that agreement is reached almost always at time 1.

The second result is more critical. The possible non-existence of a pure strategy equilibrium limits the possible employment of alternating-offers in real settings. Therefore the protocol should be modified (for example with the strategic delay option), or employing behavioural strategies.

In future work we plan to analyze alternating-offers with strategic delay, as well as carry on the study of incomplete information alternating-offers with two-sided uncertainty over deadlines, discount factors, and reservation prices.

REFERENCES

- [1] P. C. Cramton, L. M. Ausubel, and R. J. Deneckere, *Handbook of Game Theory*, volume 3, 1897–1945, Elsevier Science, 2002.
- [2] F. Di Giunta and N. Gatti, ‘Bargaining in-bundle multiple issues in finite-horizon alternating-offers’, in *Proceedings of AIMATH*, Fort Lauderdale, USA, (January 4–6 2006).
- [3] P. Faratin, C. Sierra, and N. R. Jennings, ‘Negotiation decision functions for autonomous agents’, *Robotic Autonomous Systems*, **24**(3–4), 159–182, (1998).
- [4] S. S. Fatima, M. Wooldridge, and N. R. Jennings, ‘An agenda-based framework for multi-issue negotiation’, *Artificial Intelligence*, **152**, 1–45, (2004).
- [5] D. Fudenberg and J. Tirole, *Game Theory*, The MIT Press, Cambridge, MA, USA, 1991.
- [6] J. C. Harsanyi and R. Selten, ‘A generalized Nash solution for two-person bargaining games with incomplete information’, *Management Science*, **18**, 80–106, (1972).
- [7] S. Kraus, *Strategic Negotiation in Multiagent Environments*, MIT Press, Cambridge, USA, 2001.
- [8] D. R. Kreps and R. Wilson, ‘Sequential equilibria’, *Econometrica*, **50**(4), 863–894, (1982).
- [9] S. Napel, *Bilateral Bargaining: Theory and Applications*, Springer-Verlag, Berlin, Germany, 2002.
- [10] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter. Designing Conventions for Automated Negotiations among Computers*, MIT Press, Cambridge, USA, 1994.
- [11] A. Rubinstein, ‘Perfect equilibrium in a bargaining model’, *Econometrica*, **50**(1), 97–109, (1982).
- [12] A. Rubinstein, ‘A bargaining model with incomplete information about time preferences’, *Econometrica*, **53**(5), 1151–1172, (1985).
- [13] A. Rubinstein, ‘Comments on the interpretations of game theory’, *Econometrica*, **59**(4), 909–924, (1991).
- [14] T. Sandholm, ‘Agents in electronic commerce: Component technologies for automated negotiation and coalition formation’, *Autonomous Agents and Multi-Agent Systems*, **3**(1), 73–96, (2000).
- [15] Tuomas W. Sandholm, ‘Distributed rational decision making’, in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed., Gerhard Weiss, 201–258, The MIT Press, Cambridge, MA, USA, (1999).
- [16] I. Stahl, *Bargaining Theory*, Stockholm School of Economics, Stockholm, Sweden, 1972.

A logic-based framework to compute Pareto agreements in one-shot bilateral negotiation

Azzurra Ragone and **Tommaso Di Noia** and **Eugenio Di Sciascio¹** and **Francesco M. Donini²**

Abstract. We propose a logic-based approach to automated one-shot multi-issue bilateral negotiation. We use logic in two ways: (1) a logic theory to represent relations among issues – *e.g.*, logical implication – in contrast with approaches that describe issues as uncorrelated with each other; (2) utilities over formulas to represent agents having preferences over different bundles of issues. In this case, the utility assigned to a bundle is not necessarily the sum of utilities assigned to single elements in the bundle itself. We illustrate the theoretical framework and the one-shot negotiation protocol, which makes use of a facilitator to compute some particular Pareto-efficient outcomes. We prove the computational adequacy of our method by studying the complexity of the problem of finding Pareto-efficient solutions in a propositional logic setting.

1 Introduction

Many recent research efforts have been focused on automated negotiation among agents in various contexts, including e-marketplaces, resource allocation settings, online auctions, supply chain management and, generally speaking, e-business processes. Basically, two different approaches to automated negotiation exist: *centralised* and *distributed* ones. In the first ones, agents elicit their preferences and then a facilitator, or some central entity, selects the most suitable deal based on them. In the latter ones, agents negotiate through various negotiation steps reaching the final deal by means of intermediate deals, without any external help [3]. Although the main target of an agent is reaching a satisfying agreement, knowing if it is Pareto-efficient³ is a matter that cannot be left out. Furthermore it is fundamental to assess *how hard* it is to find a Pareto-efficient agreement, and if such an agreement actually exists. Recently, there has been a growing interest toward multi-issue negotiation, also motivated by the idea that richer and expressive descriptions of demand and supply can boost e-marketplaces (*see e.g.*, [21] for a reasonable set of motivations) but –to the best of our knowledge– also in recent literature, issues are still described as uncorrelated terms, without considering any underlying semantics, which could instead be exploited –among other reasons– to perform a pre-selection of candidate deals. Moreover, agents can have different preferences over different bundles of issues, and utility assigned to each bundle is not merely the sum of utilities assigned to a single resource in that bundle. When multiple issues are involved

in the negotiation process, it is usually the case that the issues are not completely independent; sometimes an issue implies another one (for instance, *SatelliteAlarm* implies *AlarmSystem* in an automotive domain), sometimes their relation is more complex (as in bundles of optionals). We propose to fill this lack of expressiveness by relating with each other the issues involved in the process via a logical theory. In particular, here we present a framework for automated multi-issue bilateral negotiation where logic is exploited both to represent existing relations between issues and in utilities over formulas. The rest of the paper is structured as follows: next section presents a brief summary of the huge literature on bilateral negotiation. Section 3 presents the reference scenario and assumptions we make. In Section 4 the Multi-issue Bilateral Negotiation problem (MBN) is formally defined based on a logic framework. We define utility functions over a set of logic formulas representing preferences on bundles of interrelated issues. In Section 5, we prove the computational adequacy of our method by studying the complexity of the problem of finding Pareto-efficient solutions in a propositional logic setting. The bargaining process is detailed in Section 6 and an illustrative example is given in Section 7. Conclusion and future work close the paper.

2 Related Work

Automated bilateral negotiation among agents is a challenging research topic, widely investigated, both in artificial intelligence (AI) and in microeconomics research communities. Several definitions have been proposed in the literature. Rubinstein [19] defined the *Bargaining Problem* as the situation in which "two individuals have before them several possible contractual agreements. Both have interests in reaching agreement but their interests are not entirely identical. What 'will be' the agreed contract, assuming that both parties behave rationally?" Also Nash [13] defined: "a TWO-PERSON bargaining situation involves two individuals who have the opportunity to collaborate for mutual benefit in more than one way." The negotiation is *distributive*, also called Win-Lose negotiation [18], when a single item is involved and bargainers have opposite interests: if one gets more the other part gets less. If several items are negotiated simultaneously the negotiation is called *integrative* (Win-Win negotiation), each issue has a different utility (or score) for each player and differently from the distributive case, the players are not strict competitors: they can cooperate and if one gets more, the other player does not necessarily get less. Finally it is possible to have *many issues* to be negotiated among *many parties*; this is a more complex setting, because of the interplay among shifting coalitions: parties can join and act against other parties involved [17]. In game theory, the bargaining problem has been modeled either as *cooperative* or

¹ SisInflab - Technical University of Bari, Italy, email: {a.ragone, t.dinoia, di sciascio} @poliba.it

² University of Tuscia, Italy, email: donini@unitus.it

³ An agreement is Pareto-efficient if there is no other agreement that will make at least one participant better off without making at least one other participant worse off. If a negotiation outcome is not Pareto-efficient, then there is another outcome that will make at least one participant happier while keeping everyone else at least as happy [7].

non-cooperative games [6]. The first approach, given a set of axioms and a coalition, determines how to split the surplus among the participants; so the aim is finding a solution given a set of possible outcomes. Instead, in non-cooperative games there are a well-defined set of rules and strategies; in such a game it is possible to define an *equilibrium* strategy, which ensures the rational outcomes of a game: no player could benefit by unilaterally deviating from her strategy, given the other players follow their own strategies [9]. On the other hand, AI-oriented research has been more focused on automated negotiation among agents and on designing high-level protocols for agent interaction [10]. Agents can play different roles: act on behalf of buyer or seller, but also play the role of a mediator or facilitator. Agents are considered individually rational: they will never accept a deal which involves a loss, *i.e.*, a disadvantageous deal [5]. Different approaches exist to automate negotiation. Many of these consider self-interested agents negotiate over a set of resources in order to obtain an optimal allocation of such resources [5, 4, 3]. Endriss et al. [5] propose an optimal resource allocation in two different negotiation scenarios: one, with money transfer, results in an allocation with maximal social welfare; the second is a money-free framework which results in a Pareto outcome. In [3] agents negotiate over small bundles of resources and a mechanism of resource allocation is investigated, which maximizes the social welfare by means of a sequence of deals involving at most k items each. Both papers [5, 3] extend the framework proposed in [20] concerning negotiation for (re)allocating tasks among agents. We share with [22] the definition of agreement as a model for a set of formulas from both agents. However, in [22] only multiple-rounds protocols are studied, and the approach taken leaves the burden to reach an agreement to the agents themselves, although they can follow a protocol. Results do not take preferences into account, so that it is not possible to guarantee the reached agreement is Pareto-efficient. Our approach, instead, aims at giving an *automated* support to negotiating agents to reach, in one shot, Pareto agreements. In [16] an initial propositional logic framework was presented. Here we extend and generalize the approach, study the computational complexity of the problem showing how to compute a solution of the problem through combinatorial optimization techniques, and prove the computational adequacy of our method.

3 Scenario and assumptions

We illustrate our logic-based negotiation framework in accordance with [8], which defines: the *Space of possible deals*, that is the set of potential agreements; the *Negotiation Protocol*, the set of rules that an agent follows in order to reach an agreement; the *Negotiation Strategy* that an agent adopts given the set of rules specified in the negotiation protocol. In our framework the *Space of possible deals* is computed with the aid of a logic language. The *Negotiation Protocol* we adopt here is a *one-shot* protocol. We refer to *Single-shot* bargaining [17], also called *ultimatum game*, where one player proposes a deal and the other player may only accept or refuse it [2]. Therefore we focus our initial investigation on a one-shot protocol, although several other negotiation protocols exist, like *e.g.*, *alternate-offers* protocol [19] or *monotonic concession* protocol [8]. In some negotiations the parties cannot reach an agreement without some external help, that is without the intervention of a third-party. The third-party can be defined as a *facilitator*, *mediator* or *arbitrator*, depending on her influence and power in the negotiation, that is depending on her role: the more the role is evaluative, more appropriate is to define the third-party intervention as arbitration rather than facilitation or

mediation. We adopt in this paper the centralized approach, so we hypothesize the presence of an electronic facilitator, who may automatically explore the negotiation space and discover Pareto-efficient agreements to be proposed to both parties. The presence of a facilitator and the one-shot protocol is an incentive for the two parties to reveal the true preferences, because they can trust in the facilitator and they have a single possibility to reach the agreement with that counterpart. Usually bargainers may not want to disclose their preferences or utility function to the other party, but they can be ready to reveal these information to a trusted – automated – facilitator helping negotiating parties to achieve efficient and equitable outcomes [18, p.311]. Therefore we propose a one-shot protocol with the intervention of a *facilitator* with a proactive behavior: it suggests to each participant the solution which is Pareto-efficient. For what concerns *strategy*, the players can choose to accept or refuse the solution proposed by facilitator; they refuse if they think possible to reach a better agreement looking for another partner, or another shot, or for a different set of bidding rules. Here we do not consider the influence of the *outside options* in the negotiation strategy [12]. Obviously, in e-marketplaces, both buyer and seller know that the opponent is probably not the only partner available for negotiating with, and that there might be more than one single e-marketplace.

4 Logic-based Multi Issue Bilateral Negotiation

We use propositional formulas to model the buyer's demand and the seller's supply. Relations among issues are represented by a set \mathcal{T} – for Theory – of propositional formulas. We point out that each of the following definitions can be reformulated and still keep its validity within First Order Logic as long as only closed formulas are used.

Given a set $\mathcal{A} \doteq \{A_1, \dots, A_n\}$ of propositional atoms (Attributes), we denote with $\mathcal{L}_{\mathcal{A}}$ the set of propositional formulas (denoted by greek letters β, σ, \dots) built from \mathcal{A} using $\wedge, \vee, \neg, \Rightarrow$. A *propositional theory* is a finite subset of $\mathcal{L}_{\mathcal{A}}$, which we denote by $\mathcal{T} \subset \mathcal{L}_{\mathcal{A}}$. An interpretation m assigns values *true*, *false* to elements of \mathcal{A} , and we extend m to $\mathcal{L}_{\mathcal{A}}$ according to truth tables for connectives. We denote with $m \models \beta$ the fact that m assigns *true* to $\beta \in \mathcal{L}_{\mathcal{A}}$. A *model* of \mathcal{T} is an interpretation m assigning *true* to all formulas of \mathcal{T} , written $m \models \mathcal{T}$. A theory is *satisfiable* if it has a model. \mathcal{T} logically implies a formula φ , denoted by $\mathcal{T} \models \varphi$ iff φ is true in all models of \mathcal{T} . We denote with $\mathcal{M}_{\mathcal{T}} = \{m_1, \dots, m_n\}$, the set of all models for \mathcal{T} , and omit the subscript when no confusion arises.

Usually, in a bilateral negotiation the issues within both the buyer's request and the seller's offer can be split into *strict requirements* and *preferences*. The former are constraints the buyer and the seller want to be necessarily satisfied to accept the final agreement – this is what we call *demand/supply* – while the latter are issues they may accept to negotiate on – this is what we call *preferences*.

Anticipating our illustrative example, let us consider an automotive domain and suppose the buyer is asking for a sedan with leather seats, preferably red with car CD reader. She strictly wants a sedan with leather seats and expresses her preferences (she is willing to negotiate on) on the color red and the car CD reader in order to reach an agreement with the seller.

Definition 1 (Demand, Supply, Agreement) Let \mathcal{A} be a set of propositional atoms, and $\mathcal{T} \subset \mathcal{L}_{\mathcal{A}}$.

- a buyer's demand is a propositional formula β (for Buyer) in $\mathcal{L}_{\mathcal{A}}$ such that $\mathcal{T} \cup \{\beta\}$ is satisfiable.

- a seller's supply is a propositional formula σ (for Seller) in $\mathcal{L}_\mathcal{A}$ such that $\mathcal{T} \cup \{\sigma\}$ is satisfiable.
- m is a possible deal between β and σ iff $m \models \mathcal{T} \cup \{\sigma, \beta\}$, that is, m is a model for \mathcal{T} , σ , and β . We also call m an agreement.

Intuitively, σ and β stand for the minimal requirements that each partner accepts for the negotiation. On the other hand, if seller and buyer have settled strict attributes that are in conflict with each other, that is $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}} = \emptyset$, the negotiation ends immediately because, *sic stantibus rebus*, it is impossible to reach an agreement. If the participants are willing to avoid the *conflict deal* [8], and continue the negotiation, it will be necessary they revise their strict requirements.

In the negotiation process both the buyer and the seller set some preferences on attributes, or combination of attributes, they want to negotiate on. The utility function is usually defined based on these attributes. We start defining buyer's and seller's preferences and the associated utilities. Obviously, in a negotiation process there are two utility functions: u_β for the buyer, and u_σ for the seller.

Definition 2 (Preferences) Let \mathcal{A} be a set of propositional atoms, and $\mathcal{T} \subset \mathcal{L}_\mathcal{A}$.

- the buyer's negotiation preferences $\mathcal{B} \doteq \{\beta_1, \dots, \beta_k\}$ are a set of propositional formulas in $\mathcal{L}_\mathcal{A}$, each of them representing the subject of a buyer's preference, and a utility function $u_\beta : \mathcal{B} \rightarrow \mathbb{R}^+$ assigning a utility to each formula, such that $\sum_i u_\beta(\beta_i) = 1$.
- analogously, the seller's negotiation preferences $\mathcal{S} \doteq \{\sigma_1, \dots, \sigma_h\}$ are a set of propositional formulas in $\mathcal{L}_\mathcal{A}$, and a utility function $u_\sigma : \mathcal{S} \rightarrow \mathbb{R}^+$ such that $\sum_j u_\sigma(\sigma_j) = 1$.

Buyer's/seller's preferences are used to evaluate how good is a possible agreement and to select the best one. As usual, both agents' utilities are normalized to 1 to eliminate outliers, and make them comparable. Since we assumed that utilities are additive, the *global utility* is just a sum of the utilities of preferences satisfied in the agreement. To lower the number of symbols, we use u_β and u_σ also to denote the functions computing the global utility of agents over an agreement.

Definition 3 (Global Utilities) Let \mathcal{B} and \mathcal{S} be respectively the buyer's and seller's preferences, and \mathcal{M} be their agreements set. The global utility of an agreement $m \in \mathcal{M}$ for a buyer and a seller, respectively, are defined as:

$$u_\beta(m) \doteq \sum_i u_\beta(\beta_i) \mid m \models \beta_i \}, \quad u_\sigma(m) \doteq \sum_j u_\sigma(\sigma_j) \mid m \models \sigma_j \}$$

where $\sum \{\dots\}$ stands for the sum of all elements in the set.

Based on the previous definitions it is possible now to define a Multi-issue Bilateral Negotiation problem. The only other elements we still need to introduce are the *disagreement thresholds*, also called disagreement payoffs, or reservation values, t_β, t_σ , that are the minimum utility that each agent requires to pursue a deal. They may incorporate an agent's attitude toward concluding the transaction, but also overhead costs involved in the transaction itself, e.g., fixed taxes.

Definition 4 (MBN) Given a demand β and a set of buyer's preferences \mathcal{B} with utility function u_β , a supply σ and a set of seller's preferences \mathcal{S} with utility function u_σ , a propositional theory \mathcal{T} and two disagreement thresholds t_β, t_σ , a **Multi-issue Bilateral Negotiation problem (MBN)** is finding a model m (agreement) such that $m \models \mathcal{T} \cup \{\sigma, \beta\}$ and both $u_\beta(m) \geq t_\beta$ and $u_\sigma(m) \geq t_\sigma$.

Observe that not every agreement m is a solution of an MBN, if either $u_\sigma(m) < t_\sigma$ or $u_\beta(m) < t_\beta$. Such an agreement represents

a deal which, although satisfying strict requirements, it is not worth the transaction effort.

Obviously among all possible agreements that we can compute given a theory \mathcal{T} , we are interested in agreements that are Pareto-efficient, so we define:

Definition 5 (Pareto) A Pareto agreement for an MBN is an agreement m , such that for no agreement m' both $u_\beta(m') \geq u_\beta(m)$ and $u_\sigma(m') \geq u_\sigma(m)$, with at least one strict inequality.

We can also define the search of particular agreements for an MBN, maximizing the welfare $u_\beta + u_\sigma$, or maximizing the product $u_\beta \cdot u_\sigma$

Definition 6 Given an MBN, we define the following problems:

- MAX-SUM-MBN is the problem of finding an agreement m for which $u_\sigma(m) + u_\beta(m)$ is maximal
- MAX-PROD-MBN is the problem of finding an agreement m for which $u_\sigma(m) \cdot u_\beta(m)$ is maximal

Clearly, every solution for MAX-SUM-MBN and MAX-PROD-MBN is also a Pareto agreement, but not vice versa.

5 Computational issues

In this section we start by proving membership in NPO [1] of MAX-SUM-MBN and MAX-PROD-MBN. Then, we outline how to compute solutions for MAX-SUM-MBN and MAX-PROD-MBN through general combinatorial optimization techniques. Finally, we prove NPO-hardness of both problems, that is, tailored algorithms yielding solutions approximated within a polynomial bound from the optimum are unlikely to exist.

Theorem 1 Given an instance of MBN, deciding whether it has a solution can be checked in nondeterministic polynomial-time, even if a lower bound $k > 0$ is given on either the sum or the product of u_σ, u_β .

Proof. Simply guess an interpretation m , and check in polynomial time that it is a model of $\mathcal{T} \cup \{\sigma, \beta\}$ and that $u_\sigma(m) \geq t_\sigma, u_\beta(m) \geq t_\beta$. If a lower bound k is given, also checking that either $u_\sigma(m) + u_\beta(m) \geq k$ or $u_\sigma(m) \cdot u_\beta(m) \geq k$ can be done in polynomial time. \square

Hence both MAX-SUM-MBN and MAX-PROD-MBN belong to NPO. We now outline how an actual solution to both problems can be found by Integer Linear Programming (ILP).

Regarding MAX-SUM-MBN, proceed as follows. First of all, let $\{B_1, \dots, B_k, S_1, \dots, S_h\}$ be $k+h$ new propositional atoms, and let $\mathcal{T}' = \mathcal{T} \cup \{B_i \equiv \beta_i \mid i = 1, \dots, k\} \cup \{S_j \equiv \sigma_j \mid j = 1, \dots, h\}$ – that is, every preference in $\mathcal{B} \cup \mathcal{S}$ is equivalent to a new atom in \mathcal{T}' . Then, by using standard transformations in clausal form [11] obtain a set of clauses \mathcal{T}'' which is satisfiable iff $\mathcal{T}' \cup \{\sigma, \beta\}$ does. Then, use a well-known encoding of clauses into linear disequations (e.g., [14, p.314]) so that every solution of the disequations identifies a model of \mathcal{T}'' . Let $\{b_1, \dots, b_k\}$ the (0,1)-variables one-one with $\{B_1, \dots, B_k\}$ and similarly $\{s_1, \dots, s_h\}$. Then, maximize the linear function

$$\sum_{i=1}^k b_i u_\beta(\beta_i) + \sum_{j=1}^h s_j u_\sigma(\sigma_j)$$

Regarding MAX-PROD-MBN, let $\{B_{ij} \mid i = 1, \dots, k, j = 1, \dots, h\}$ be a set of $k \cdot h$ new propositional atoms. Let $\mathcal{T}' = \mathcal{T} \cup \{B_{ij} \equiv$

$\beta_i \wedge \sigma_j \mid i = 1 \dots, k \ j = 1, \dots, h\}$ – that is, this time every pair of conjoined preferences is equivalent to a new atom. Then, transform \mathcal{T}' into a set of clauses \mathcal{T}'' , code clauses as disequations and maximize

$$\sum_{i=1}^k \sum_{j=1}^h b_{ij} u_\beta(\beta_i) u_\sigma(\sigma_j)$$

Hence, also MAX-PROD-MBN can be solved by ILP as a problem whose size is at most quadratic in the size of the original MAX-PROD-MBN.

When an optimization problem is NP-complete, one can ask whether there exists an algorithm approximating the optimum. For instance, it is known that the following problem MAX-WEIGHTED-SAT is not approximable [1] within any polynomial.

Definition 7 MAX-WEIGHTED-SAT is the following NPO-complete problem: given set of atoms \mathcal{A} , a propositional formula $\varphi \in \mathcal{L}_\mathcal{A}$ and a weight function $w : \mathcal{A} \rightarrow \mathbb{N}$, find a truth assignment satisfying φ such that the sum of the weights of true variables is maximum.

If MAX-SUM-MBN or MAX-PROD-MBN admitted some approximation algorithm, using general ILP for solving them might be an overshoot. However, the theorem below proves that this is not the case, by giving an L-reduction [1] of MAX-WEIGHTED-SAT to both problems.

Theorem 2 MAX-SUM-MBN and MAX-PROD-MBN are NPO-complete problems, even if \mathcal{T} is in 3CNF and both \mathcal{B} and \mathcal{S} are sets of positive literals.

Proof. Let $W = \langle \varphi, w \rangle$ be an instance of MAX-WEIGHTED-SAT, with φ in 3CNF. Define an instance M_+ of MAX-SUM-MBN as follows. Let $\mathcal{T} = \varphi$, $\mathcal{B} = \{A_1, \dots, A_k\}$ with any $k = 1, \dots, n - 1$, and let $\mathcal{S} = \{A_{k+1}, \dots, A_n\}$. Moreover, let $u_\beta(A_i) = w(A_i)$ for $i = 1, \dots, k$ and $u_\sigma(A_j) = w(A_j)$ for $j = k + 1, \dots, n$. Finally, let $t_\beta = t_\sigma = 0$. Clearly, every solution for W is also a solution for M_+ , and for every model m , the value of the objective function $u_\beta(m) + u_\sigma(m)$ is the same as the one for W . Hence, the above is an L-reduction with $\alpha = \beta = 1$.

Similarly, define an instance M_\times of MAX-PROD-MBN as follows. Let $\mathcal{T} = \varphi$, $\mathcal{B} = \mathcal{A}$, $\mathcal{S} = \{A_0\}$ where A_0 is a new atom not in \mathcal{A} . Moreover, let $u_\beta(A_i) = w(A_i)$ for $i = 1, \dots, n$ and $u_\sigma(A_0) = 1$. Finally, let $t_\beta = t_\sigma = 0$. Also in this case, every solution for W satisfies also M_\times , and for every model m the objective function $u_\beta(m) \cdot u_\sigma(m)$ of M_\times has the same value as W 's one. Hence, also the above reduction is an L-reduction with $\alpha = \beta = 1$. \square

The proof of the above theorem highlights the fact that a source of complexity for MAX-SUM-MBN and MAX-PROD-MBN comes from the inherent conflicts between the preferences of a single agent.

6 The bargaining process

We now present the bargaining process, which covers the following phases:

Preliminary Phase. The buyer defines β and \mathcal{B} , as well as the threshold t_β , and the seller defines σ , \mathcal{S} and t_σ . In this paper we are not interested in how to compute these values; we assume they are determined in advance by means of either direct assignment methods (Ordering, Simple Assessing or Ratio Comparison) or pairwise comparison methods (like AHP and Geometric Mean) [15]. Both agents inform the facilitator about these specifications and the theory \mathcal{T} they refer to.

Initial phase. The facilitator computes $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$. If $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}} = \emptyset$ the facilitator requests the buyer and the seller to refine –respectively– their β and σ and recomputes $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$. Once/If buyer and seller solve any conflicts about reservation requirements, the negotiation can start.

Negotiation-Core phase. The facilitator proposes one of the two agreements mutually beneficial for both the buyer and the seller: an agreement which maximizes the social welfare and one which maximizes the product of utilities. If either one of the participants rejects it, then the facilitator proposes the second one. Since both of the agreements are Pareto-efficient, the order they are proposed to the participants is not relevant. From this point on, it is a *take-it-or-leave-it* offer, because the participants can either accept or reject it [7]. Notice that since both the proposed deals are Pareto-efficient, although the participants can move from the proposed deals, they might end in a less (no-Pareto) efficient agreement. Once the facilitator computes the space of all possible deals $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$, it evaluates the utility of each model $m \in \mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$, both for the buyer – $u_\beta(m)$ – and for the supplier – $u_\sigma(m)$.

Among all possible models, it chooses \bar{m} maximizing the product $u_\beta(m) * u_\sigma(m)$ and \underline{m} maximizing the sum. The facilitator proposes such solutions to the opponents. If one of the opponents, or both of them, refuse the deals proposed by the facilitator, they reach a *conflict deal* and then they will likely look for other partners in the e-marketplace.

7 An illustrative example

In the following we illustrate our logic based approach to automate negotiation process with the aid of a simple example, which refers to an e-marketplace related to the automotive domain. Buyer and seller express their demand/supply and preferences. According to the previous definitions, a simple instance of an MBN can be formalized as follows:

$$\begin{aligned} \beta &= \text{StationWagon} \wedge \text{AirConditioning} \wedge \text{DriverAirbag} \wedge \text{SatelliteAlarm} \\ \beta_1 &= \neg \text{ExternalColorRed} \wedge \text{ExternalColorBlack} \wedge \text{InteriorColorBlack} \\ \beta_2 &= \text{Gasoline} \vee \text{Diesel} \\ u_\beta(\beta_1) &= 0.6 \\ u_\beta(\beta_2) &= 0.4 \\ t_\beta &= 0.3 \\ \\ \sigma &= \text{StationWagon} \wedge \text{Diesel} \\ \sigma_1 &= (\text{ExternalColorBlack} \vee \text{ExternalColorBlue}) \wedge (\text{InteriorColorGrey} \vee \text{InteriorColorBlack}) \\ \sigma_2 &= \text{DriverAirbag} \wedge \neg \text{PassengerAirbag} \\ \sigma_3 &= \text{AlarmSystem} \wedge \neg \text{SatelliteAlarm} \\ u_\sigma(\sigma_1) &= 0.2 \\ u_\sigma(\sigma_2) &= 0.3 \\ u_\sigma(\sigma_3) &= 0.5 \\ t_\sigma &= 0.2 \end{aligned}$$

In this example the thresholds are set considering some transaction costs like *e.g.*, operating costs for the seller and cost of taxes for the buyer.

$$\mathcal{T} = \left\{ \begin{array}{l} \text{ExternalColorBlue} \Rightarrow \neg \text{ExternalColorBlack} \\ \text{ExternalColorBlue} \Rightarrow \neg \text{ExternalColorRed} \\ \text{ExternalColorBlack} \Rightarrow \neg \text{ExternalColorRed} \\ \text{InteriorColorBlack} \Rightarrow \neg \text{InteriorColorGrey} \\ \text{Gasoline} \Rightarrow \neg \text{Diesel} \\ \text{SatelliteAlarm} \Rightarrow \text{AlarmSystem} \\ \text{PassengerAirbag} \Rightarrow \text{FrontAirbag} \\ \text{DriverAirbag} \Rightarrow \text{FrontAirbag} \end{array} \right.$$

ExBlue	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ExBlack	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
ExRed	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
InBlack	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0
InGrey	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0
Gas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dies	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SatAlarm	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Alarm	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
PassAir	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
FrontAir	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
DrivAir	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
StatWag	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
AirCond	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
β_1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
β_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$u_\beta(m)$	0.4	0.4	0.4	0.4	0.4	0.4	1	1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
σ_1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
σ_2	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
σ_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$u_\sigma(m)$	0.2	0.5	0.2	0.5	0	0.3	0.2	0.5	0.2	0.5	0	0.3	0	0.3	0	0.3	0	0.3	0	0.3	0	0.3	0
SUM	0.6	0.9	0.6	0.9	0.4	0.7	1.2	1.5	0.6	0.9	0.4	0.7	0.4	0.7	0.4	0.7	0.4	0.7	0.4	0.7	0.4	0.7	0.7
PROD	0.08	0.2	0.08	0.2	0	0.12	0.2	0.5	0.08	0.2	0	0.12	0	0.12	0	0.12	0	0.12	0	0.12	0	0.12	0.12

Table 1. $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$ and the utility associated to each model. In bold style the values chosen by the facilitator maximizing the sum and the product.

Table 1 shows all the models in the set $\mathcal{M}_{\mathcal{T} \cup \{\sigma, \beta\}}$. As it is shown in Table 1, after computing $u_\beta(m)$ and $u_\sigma(m)$ (rows 18 and 23) the sum and the product of the two utilities are calculated. The facilitator proposes to the participants (buyer and seller) the deal that maximizes the product and the sum of utilities (in this case the same). In the final agreement we have the following propositional atoms set true:

$m = \overline{m} = \text{StationWagon, AirConditioning, DriverAirbag, SatelliteAlarm, Diesel, AlarmSystem, FrontAirbag, ExternalColorBlack, InteriorColorBlack}$

where bold style atoms highlight strict attributes, while the others are preferences. The utilities related to \overline{m} are:

$$\begin{aligned} u_\beta(\overline{m}) &= u_\beta(\beta_1) + u_\beta(\beta_2) = 0.6 + 0.4 = 1 \\ u_\sigma(\overline{m}) &= u_\sigma(\sigma_1) + u_\sigma(\sigma_2) = 0.2 + 0.3 = 0.5 \end{aligned}$$

while product and sum are:

$$\begin{aligned} u_\beta(\overline{m}) * u_\sigma(\overline{m}) &= 1 * 0.5 = 0.5 \\ u_\beta(\overline{m}) + u_\sigma(\overline{m}) &= 1 + 0.5 = 1.5 \end{aligned}$$

8 Conclusion and Future Work

In this paper we proposed a logic based approach to automate multi issue bilateral negotiation (MBN), defining a protocol which leads to a Pareto-efficient deal. The relations among issues are represented through a logic theory \mathcal{T} and preferences are elicited by assigning utility to formulas, and then considering bundles of interrelated issues. Computational issues were also investigated: we proved that MAX-SUM-MBN and MAX-PROD-MBN are NPO-complete problems. In the near future, we plan to extend the approach using Description Logics, to cope with greater expressiveness in demand/supply descriptions. We are also investigating other negotiation protocols –also different from one-shot ones– still suitable for e-commerce applications. A prototype is also being implemented to validate the approach through large scale experiments.

REFERENCES

- [1] G. Ausiello, P. Crescenzi, V. Kann, G. Gambosi, and A. M. Spaccamela. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 2003.
- [2] K. Binmore. *Fun and Games. A Text on Game Theory*. D.C. Heath and Company, 1992.
- [3] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Negotiating over small bundles of resources. In *Proc. of AAMAS '05*, pages 296–302, 2005.
- [4] P. E. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artif. Intell.*, 164(1-2):23–46, 2005.
- [5] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations over resources. In *Proc. of AAMAS '03*, pages 177–184, 2003.
- [6] E. H. Gerding, D. D. B. van Bragt, and J. A. L. Poutre. Scientific approaches and techniques for negotiation: a game theoretic and artificial intelligence perspective. Technical report, SEN-R0005, CWI, 2000.
- [7] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. Automated negotiation: prospects, methods and challenges. *Int. J. of Group Decision and Negotiation*, 10(2):199 – 215, 2001.
- [8] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press., 1994.
- [9] C. Li, J. Giampapa, and K. Sycara. A review of research literature on bilateral negotiations. Technical report, Carnegie Mellon University, 2003.
- [10] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Int Journal of Group Decision and Negotiation*, 12 (1):31–56, 2003.
- [11] D. W. Loveland. *Automated theorem proving: A logical basis*. North-Holland, 1978.
- [12] A. Muthoo. On the strategic role of outside options in bilateral bargaining. *Operations Research*, 43(2):292–297, 1995.
- [13] J. F. Nash. The bargaining problem. *Econometrica*, 18 (2):155–162, 1950.
- [14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.
- [15] J. Pomerol and S. Barba-Romero. *Multicriterion Decision Making in Management*. Kluwer Series in Operation Research. Kluwer Academic, 2000.
- [16] A. Ragone, T. Di Noia, E. Di Sciascio, and F. M. Donini. Propositional-logic approach to one-shot multi issue bilateral negotiation. *ACM SIGecom Exchanges*, 5(5):11–21, 2006.
- [17] H. Raiffa. *The Art and Science of Negotiation*. Harvard University Press, 1982.
- [18] H. Raiffa, J. Richardson, and D. Metcalfe. *Negotiation Analysis - The Science and Art of Collaborative Decision Making*. The Belknap Press of Harvard University Press, 2002.
- [19] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50:97–109, 1982.
- [20] T. Sandholm. Contract types for satisfying task allocation: I theoretical results. In *Proceedings of the AAAI Spring Symposium*, 1998.
- [21] D. Trastour, C. Bartolini, and C. Priest. Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *Proc. WWW '02*, pages 89–98, 2002.
- [22] M. Wooldridge and S. Parsons. Languages for negotiation. In *Proc of ECAI '04*, pages 393–400, 2000.

Towards ACL semantics based on commitments and penalties

Leila Amgoud and Florence Dupin de Saint-Cyr¹

Abstract.

The importance of defining a standard framework for agent communication languages (ACL) with a simple, clear, and a verifiable semantics has been widely recognized.

This paper proposes a *logic-based* semantics which is *social* in nature. The basic idea is to associate with each speech act a meaning in terms of the *commitment* induced by that speech act, and the *penalty* to be paid in case that commitment is violated. A *violation criterion* based on the existence of arguments is then defined per speech act. Moreover, we show that the proposed semantics satisfies some key properties that ensure the approach is well-founded. The logical setting makes the semantics verifiable.

1 Introduction

When building multi-agent systems, we take for granted the fact that agents which make up the system will need to engage in the different types of dialogues identified by Walton and Krabbe in [11], using a communication language (ACL). The definition of an ACL from the syntactic point of view (the different *speech acts*² that agents can perform during a dialogue) poses no problems. The situation is different when semantics is taken into account. Given that agents in a multi-agent system may be independently designed by different programmers, a clear understanding of semantics is essential. Indeed, any speech act should have a *unique interpretation*. Moreover, it should be *verifiable*, i.e. it should be possible to check whether a system conforms to a particular ACL or not [13]. Although a number of agent communication languages have been developed, obtaining a suitable formal semantics for ACLs which satisfies the above objective remains one of the greatest challenges of multi-agent theory.

Our aim is to define a semantics which prevents the shortcomings of existing approaches while keeping their benefits. The basic idea behind our semantics is that each speech act has a goal. For instance, behind a question, one expects an answer. Hence, during a dialogue, as soon as a speech act is uttered, a kind of *commitment* for achieving its goal is created. In the case of a question, by uttering such a speech act, a commitment for answering is created (here for the hearer). Note that this does not mean at all that the hearer should necessarily answer.

The new semantics is grounded on a computational logic framework, thus allowing automatic verification of compliance by means of proof procedures. More precisely, the semantics associates with each speech act a meaning in terms of the *commitment* induced by it, and a *penalty* to be paid in case that commitment is violated. A *vi-*

lation criterion based on the existence of arguments is then defined per speech act.

From a *syntactic* point of view, utterances are stored in *commitment stores* as in [7]. Each agent is supposed to be equipped with a commitment store visible to all agents.

Note that the aim of this paper is not to propose a dialogue protocol whose role is to ensure coherent dialogues, etc. However, the protocol of a dialogue system that uses our semantics should at least enforce agents to minimize their violation costs. The definition of such protocol is beyond the scope of this paper.

This paper is organized as follows: section 2 introduces the logical language used throughout the paper. Section 3 defines the new semantics. Some logical properties are presented in section 4, and an example is given in section 5. Section 6 compares our semantics with existing approaches.

2 The logical language

Throughout the paper, we consider a *propositional language* \mathcal{L} . \vdash denotes classical inference and \equiv logical equivalence. A knowledge base Σ is a set of formulas of \mathcal{L} . *Arguments* can be built from any knowledge base Σ :

Definition 1 (Argument) An argument is a pair (S, c) where c is a formula of \mathcal{L} and $S \subseteq \Sigma$ such that:

1. S is consistent,
2. $S \vdash c$,
3. S is minimal for set inclusion among the sets satisfying 1) and 2).

S is the support of the argument and c its conclusion. $\text{Arg}(\Sigma)$ is the set of all the arguments that can be built from Σ .

Given that a knowledge base Σ may be inconsistent, arguments may be conflicting too. In what follows we will use the “undercut” relation which is the most suitable in our case.

Definition 2 (Undercut) Let $A_1 = (H_1, c_1)$, $A_2 = (H_2, c_2) \in \text{Arg}(\Sigma)$. A_1 undercuts A_2 if $\exists h'_2 \in H_2$ such that $c_1 \equiv \neg h'_2$.

Let $\mathcal{A} = \{ag_1, \dots, ag_n\}$ be the set of *agents* involved in the system. Each agent is assumed to have a *role* allowing it to have the control over a subset of formulas in \mathcal{L} (Role: $\mathcal{A} \mapsto 2^{\mathcal{L}}$). Roles are supposed to be visible to all agents. Thus they are not private.

A communication language is based on a set of *speech acts*. Let \mathcal{S} denote that set. From \mathcal{S} and \mathcal{L} , different moves can be built.

Definition 3 (Move) If $a \in \mathcal{S}$ and either $x \in \mathcal{L}$ with $x \not\vdash \perp$, or $x \in \text{Arg}(\mathcal{L})$ then $a:x$ is a move.

¹ IRIT - CNRS 118, route de Narbonne 31062 Toulouse Cedex 9, France
 amgoud, bannay@irit.fr

² The speech acts are also called *illocutionary acts* or *performatives*.

For a given move $a : x$, the function **Act** returns the speech act ($\text{Act}(a : x) = a$), and the function **Content** returns the content of the move ($\text{Content}(a : x) = x$). Let \mathcal{M} denote the set of all the possible moves that can be built from S and \mathcal{L} .

For example, the move **Question**: x (with x meaning that the sky is blue) is uttered to ask whether the sky is blue or not.

3 Semantics

The basic idea behind our semantics is to associate with each speech act a meaning in terms of the *commitment* induced by that speech act, and a *penalty* to be paid in case that commitment is violated. For each speech act, we define a *criterion* precising when the corresponding commitment is violated. These criteria are all based on the existence of arguments. The various moves uttered during a dialogue are stored in *commitment stores* which are visible to all agents.

3.1 Speech acts

We consider the following set of basic speech acts that are used in the literature for modeling the different types of dialogues identified by Walton and Krabbe in [11]:

$$\mathcal{S} = \{\text{Assert}, \text{Argue}, \text{Declare}, \text{Question}, \text{Request}, \text{Challenge}, \text{Promise}\}.$$

- **Assert** and **Argue** allow an agent to inform another agent about the state of the world. These acts are *assertive* according to the classification of Searle [9]. **Assert**: x and **Argue**: x differ in the syntactic form of x . In the case of **Assert**: x , x is a proposition ($x \in \mathcal{L}$) like “the weather is beautiful” or “it is my intention to hang a mirror”. However, in **Argue**: x , x is an argument ($x \in \text{Arg}(\mathcal{L})$).

- **Declare**: x , with $x \in \mathcal{L}$, is a move that brings about a state of affairs that makes its content x true. It is a *declarative* act like “the auction is open” or “John and Mary are husband and wife”.

- **Question**, **Request** and **Challenge** are *directive* acts that incite the agent who receives them to give an answer. The moves **Question**: x , **Request**: x and **Challenge**: x are all about the same kind of information, x is a *proposition* in the three cases ($x \in \mathcal{L}$). In **Question**: x , the agent asks for the truth value of x . In **Request**: x , the agent asks another agent to alter the value of x to true. **Request**: x has then a more imperative character than **Question**: x which does not ask the other agent to act on the world but only to give some information. A **Request** is used when an agent cannot, or prefers not to, achieve one of its goals alone. For instance, if ag_2 utters **Request**: $ag_2.\text{is_paid}$ then it means that ag_2 asks to be paid. By doing a **Challenge**: x move an agent asks for a reason/argument in favor of the conclusion x .

- A **Promise** move commits the agent to some future course of action. It is a *commissive* act according to Searle [9]. The expression **Promise**: x means that the agent is committed to make x true in the future, with $x \in \mathcal{L}$. For example, if ag_1 utters **Promise**: $ag_2.\text{is_paid}$, it means that ag_1 commits itself to ensure that ag_2 is paid in the future.

- In addition to the above speech acts, we will consider another act called **Retract** which does not belong to the different categories of speech acts defined by Searle. It can be seen as a meta-level act allowing agents to withdraw commitments already made. Allowing such a move makes it possible for agents to have a kind of non-monotonic behavior (i.e., to change their points of view, to revise their beliefs, etc.) without being sanctioned. Syntactically,

Retract: m is a move with $m \in \mathcal{M}$ being itself a possible move.

3.2 Commitments

In the scientific literature, one can find proposals where the semantics of an ACL is defined in terms of commitments. Examples of these are given by Colombetti [2] and Singh [10]. Colombetti and Singh argued that agents are social entities, involved in social interactions, so they are committed to what they say. In recent inter-agent communication approaches, the notions of dialogue games and (social) commitments are central. One rather influential dialogue game is DC, proposed by Hamblin [5]. DC associates with each player a *commitment store*, which holds its commitments during the dialogue. Commitments here are pieces of information given by players during the dialogue. Then, there are rules which define how commitment stores are updated. Take for instance assertion, it puts a propositional statement in the speaker’s commitment store. What this basically means is that, when challenged, the speaker will have to justify his claim. But this does not presuppose that the challenge will come at the next turn in the dialogue.

We adopt this representation of commitments. Note that in this paper we are not interested in modeling the reasoning of the agent, we only consider what is said by each agent. The idea is to provide a semantics without worrying about agents mental states. Each agent is supposed to be equipped with a commitment store, accessible to all agents, that will contain the utterances it makes during the dialogue. A commitment store keeps tracks of two kinds of speech acts:

- Speech acts made by the agent itself such as assertions, promises and declarations.
- Speech acts received from other agents, such as requests, challenges and questions. For instance if an agent ag_i makes a request r to another agent ag_j , the request (r) is stored in the commitment store of ag_j . Hence, ag_j is said *committed* to answer to it.

Definition 4 (Commitment store) A commitment store CS_i associated with an agent ag_i is a pair $CS_i = \langle A_i, O_i \rangle$ with:

$$\begin{aligned} A_i &\subseteq \{m \in \mathcal{M} \mid \text{Act}(m) \in \{\text{Assert}, \text{Argue}, \text{Declare}, \text{Promise}\}\}. \\ O_i &\subseteq \{m \in \mathcal{M} \mid \text{Act}(m) \in \{\text{Question}, \text{Request}, \text{Challenge}\}\}. \end{aligned}$$

A dialogue evolves from one step to another as soon as a move is uttered. In what follows, CS_i^s denotes the commitment store of agent i at step s . A commitment store is supposed to be *empty* at the beginning of a dialogue (i.e., at step 0). Hence, for all agent ag_i , $CS_i^0 = \emptyset$. Given a set X of moves, X^i denotes the moves of X that are uttered from step 0 to step i . Let us now introduce two functions **PROP** and **PROP_P** that return sets of formulas as follows:

Definition 5 Let $X \subseteq \mathcal{M}$.

- **PROP**(X) is defined recursively by:

$$\text{PROP}(X^0) = \emptyset$$

$$\text{PROP}(X^s) = \begin{cases} \text{PROP}(X^{s-1}) \cup \{x\} & \text{if } m = \text{Assert}:x \\ \text{PROP}(X^{s-1}) \cup S & \text{if } m = \text{Argue}:(S, c) \\ \text{PROP}(X^{s-1}) \diamond x & \text{if } m = \text{Declare}:x \\ \text{PROP}(X^{s-1}) & \text{else} \end{cases}$$

where m is the move uttered at step s in X^s and \diamond is an update operator described below.

- **PROP_P**(X) = { $x \in \mathcal{L}$ such that $\exists \text{Promise}:x \in X$ }.

The above definition computes the set of formulas that represent the state of the world (according to what has been uttered during the dialogue). Note that Questions, Challenges and Requests are not considered in the definition because they don't describe the state of the world. Formulas that appear in assertions and arguments are directly considered. However, things are different with the formulas related to a move **Declare**. Indeed, by definition, after **Declare**: x the world evolves in such a way that x becomes true. Consequently, one has to update the whole set of propositions previously uttered. For that purpose, an update operator [12], denoted by \diamond , is needed. Several update operators have been introduced in the literature. The choice of the precise one to be used in our semantics is beyond the scope of this paper.

3.3 The notion of penalty

As said before, from each move a commitment is induced. It is natural to associate with each commitment a penalty that sanctions the agent when this commitment is violated. For the sake of simplicity, the penalty is supposed to depend on the speech act and not on the content of the move. Hence, each speech act in \mathcal{S} is supposed to have a *cost* which is an integer: $\text{Cost} : \mathcal{S} \rightarrow \mathbb{N}$. Different speech acts may have different values. This captures the idea that some speech acts are more important than others. For instance, violating a promise may be more costly than not answering a question. With each commitment store is associated a penalty as follows:

Definition 6 (Penalty) Let $CS_i = \langle A_i, O_i \rangle$ be a commitment store, and $X \subseteq A_i \cup O_i$. The penalty associated with X w.r.t. CS_i is

$$c(X) = \sum_{m \in X} \text{Penalty}(m)$$

where $\text{Penalty}(m) = \text{Cost}(\text{Act}(m))$ if the commitment m is violated in A_i and $\text{Penalty}(m) = 0$ otherwise.

Since a commitment store is empty at the beginning of a dialogue, its initial penalty is equal to 0. Moreover, at any step, the penalty of a given commitment store can be computed in a very simple way as shown in the next section.

3.4 Violation criteria

As shown before, a penalty is to be paid if a commitment is violated. This section presents in details when commitments induced from each speech act of \mathcal{S} are violated. Subsequently, we suppose that the agent ag_i utters the move to the agent ag_j .

1. Assert: x

During a dialogue, an agent can assert that a given propositional formula is true. Then, this agent is not allowed to contradict itself during all the dialogue otherwise it will have to pay a penalty (except if it retracts that proposition). Indeed, a move **assert**: x is *violated* if the A_i part of the commitment store of the agent ag_i makes it possible to find an argument with a conclusion $\neg x$. Formally:

Definition 7 A move **Assert**: x is violated iff

$$\exists(S, \neg x) \in \text{Arg}(\text{PROP}(A_i)).$$

In order to avoid any form of wishful thinking, in the above definition, the promises are not taken into account when checking the violation of an assert move, even if they are stored in the A_i part

of the commitment store.

2. Argue: x

During a dialogue, an agent can provide an argument x in favor of some conclusion. Then, this agent is not allowed to contradict itself in the sense that it cannot produce an undercut against x .

Definition 8 A move **Argue**: x is violated iff

$$\exists(S', y) \in \text{Arg}(\text{PROP}(A_i)) \text{ such that } (S', y) \text{ undercuts}^3 x.$$

As for assert moves and for the same reason, promises are not taken into account when looking for counter arguments.

3. Declare: x

During a dialogue, an agent can modify the state of a certain proposition x by declaring it true. The move **Declare**: x commits the honesty of the agent which carries it out in the sense that the agent should be empowered to modify the value of x . This capacity is defined by the role of the agent. For instance, it is not allowed for a simple citizen to marry people. Moreover, an agent can really modify this value only if there is no argument against performing that action. Formally:

Definition 9 A move **Declare**: x is violated iff

$$x \notin \text{Role}(ag_i) \text{ or } \exists(S, \neg y) \in \text{Arg}(\text{Prop}(A_i)) \\ \text{with } y \in \text{Precond}(x)$$

where $\text{Precond} : \mathcal{L} \rightarrow 2^{\mathcal{L}}$ is a function that gives for any formula φ the pre-conditions for setting φ to true and that verifies: $\text{Precond}(\perp) = \{\perp\}$ and $\text{Precond}(\top) = \emptyset$

The definition of **Precond** may come from law, it is supposed to be furnished (its definition is beyond the scope of this paper). For example, in order to open an auction, one should check whether the buyers are present. If a formula can never be set to true then the function **Precond** returns $\{\perp\}$. When, there is no pre-condition for setting the formula to true, the function returns \emptyset .

4. Question: x

During a dialogue, an agent may receive questions from other agents to which it should answer either positively or negatively. The absence of any argument in favor of x or $\neg x$ in the part A_j of the commitment store of the agent that receives the move means that the agent has not given any answer.

Definition 10 A move **Question**: x is violated iff

- $\nexists(S, x) \in \text{Arg}(\text{PROP}(A_j))$ and
- $\nexists(S, \neg x) \in \text{Arg}(\text{PROP}(A_j))$.

Again, promises are not considered when building arguments. Note that we check the existence of an argument in favor of x or $\neg x$ instead of just the existence of a proposition equivalent to x or to $\neg x$ in A_j . The reason is that the question can be answered implicitly via other assertions of the agent. In this setting, it is not possible to answer "I don't know" to a question. But, this could be easily handled by introducing the speech act **Desinform**.

5. Request: x

An agent should give a positive or a negative answer to any request it receives from other agents.

³ See Definition 2 in Section 2.

Definition 11 A move $\text{Request}:x$ is violated iff

- $\nexists (S, x) \in \text{Arg}(\text{PROP}(A_j) \cup \text{PROP}_P(A_j))$ and
- $\nexists (S', \neg x) \in \text{Arg}(\text{PROP}(A_j) \cup \text{PROP}_P(A_j))$.

Note that to check whether a request is violated or not, we look for an argument in favor of x in both $\text{PROP}(A_j)$ and $\text{PROP}_P(A_j)$. The reason is that a request can get an answer in two ways: either because of a promise ensuring that in the future the requested proposition will be set to true or to false, or because it is already stated (either by declarations or assertions) to true or false.

6. Challenge: x

Agents should provide arguments for any challenged proposition.

Definition 12 A move $\text{Challenge}:x$ is violated iff

$$\nexists (S, x) \in \text{Arg}(\text{PROP}(A_j)) \text{ with } S \not\equiv x.$$

Let us take the example of agent which asserts x , after which the other agent makes a challenge on x . It is clear that the argument $(\{x\}, x)$ can be built from $\text{Arg}(\text{PROP}(A_j))$, however this is not an answer to the challenge. Thus in order to avoid such problem, the above definition requires that the argument presented after a challenge should be different from x .

7. Promise: x

During a dialogue, an agent can make promises to other agents. This agent should pay a penalty in case it does not respect this promise. This can be checked on the part A_i of its commitment store. Indeed, if an argument in favor of proposition x can be built then the promise is honored otherwise it is considered violated.

Definition 13 A move $\text{Promise}:x$ is violated iff

$$\nexists (S, x) \in \text{Arg}(\text{PROP}(A_i)).$$

8. Retract: m

Agents may decide to retract some previously uttered moves. The advantage of this move is to allow them to revise their beliefs without being sanctioned. The Retract move is different from the others since it is never violated, thus $\text{Penalty}(\text{Retract}:m) = 0$. Moreover, after such a move the commitment store is updated as follows:

Definition 14 Let CS_i^s be the commitment store of an agent ag_i at step s . A move $\text{Retract}(m)$ at step $s + 1$ has the following effect:

$$CS_i^{s+1} = CS_i^s \setminus \{m\}$$

Note that retracting a move that has not been uttered has no effect.

4 Logical properties

The aim of this section is to show that the proposed semantics satisfies some key and desirable properties. The first property ensures that the semantics sanctions only bad behaviors of agents, and that any bad behavior is sanctioned.

Proposition 1

- If $c(CS_i) > 0$, then $\exists m \in CS_i$ s.t m is violated.
- If $\exists m \in CS_i$ s.t m is violated, then $c(CS_i) > 0$.

Another important result is the fact that if the total penalty of part A_i is null then all the stated information is consistent.

Proposition 2 (Consistency) If $\sum_{m \in A_i} \text{Penalty}(m) = 0$, then $\text{PROP}(A_i)$ is consistent.

In [6], it has been shown that a propositional formula may be useful for explaining another formula in a given context. This property is called *novelty*. In what follows, we give its definition in terms of arguments.

Definition 15 (Novelty) Let φ, ϕ be two propositional formulas, and Σ a set of formulas.

- φ is new for ϕ w.r.t. Σ iff:
 - $\exists \langle S, \phi \rangle \in \text{Arg}(\Sigma \cup \varphi)$ and $\langle S, \phi \rangle \notin \text{Arg}(\Sigma)$, or
 - $\exists \langle S, \neg\phi \rangle \in \text{Arg}(\Sigma \cup \varphi)$ and $\langle S, \neg\phi \rangle \notin \text{Arg}(\Sigma)$
- φ is said to be independent from ϕ w.r.t. Σ otherwise.

We can show that if two formulas are independent w.r.t. the formulas of a commitment store, then the penalty of two moves conveying these formulas is decomposable. Formally:

Proposition 3 (Independence) Let $CS_i = \langle A_i, O_i \rangle$ be a commitment store, and let $m, m' \in \mathcal{M}$. If $\text{Content}(m)$ is independent from $\text{Content}(m')$ w.r.t. $\text{PROP}(A_i) \cup \text{PROP}_P(A_i)$, then

$$c(\{m, m'\}) = c(\{m\}) + c(\{m'\})$$

5 Example

Let us study the following dialogue between two agents ag_1 and ag_2 :

$ag_2 \rightarrow ag_1$: Do you think that Newspapers can publish (pub) the information X .

A_1	O_1
\emptyset	Question: pub

$c(CS_1) = \text{Cost}(\text{Question})$

$ag_1 \rightarrow ag_2$: No.

A_1	O_1
Assert: $\neg pub$	Question: pub

$c(CS_1) = 0$

$ag_2 \rightarrow ag_1$: why?

A_1	O_1
Assert: $\neg pub$	Question: pub
Argue: $(\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub)$	Challenge: $\neg pub$

$c(CS_1) = \text{Cost}(\text{Challenge})$

$ag_1 \rightarrow ag_2$: Because X concerns the private life of A (pri) and A does not agree to publish it (agr).

A_1	O_1
Assert: $\neg pub$	Question: pub
Argue: $(\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub)$	Challenge: $\neg pub$
$c(CS_1) = 0$	

$ag_2 \rightarrow ag_1$: But A is a minister (min) and information about ministers are public.

A_2	O_2
Argue: $(\{min, min \rightarrow \neg pri\}, \neg pri)$	\emptyset

$c(CS_2) = 0$

$ag_1 \rightarrow ag_2$: Yes, you are right.

A_1	O_1
Assert: $\neg pub$	Question: pub
Argue: $(\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub)$	Challenge: $\neg pub$

$c(CS_1) = 2 \times \text{Cost}(\text{Argue})$

In the above example, the agent ag_1 answers the question and the challenge it received, thus there is no penalty to pay for those moves. However, this agent has presented an argument $a = \langle \{pri, \neg agr\}, \neg pub \rangle$, and accepted its undercutter $b = \langle \{min, min \rightarrow \neg pri\}, \neg pri \rangle$. Consequently, the set $\text{PROP}(A_i)$ is inconsistent, and this makes it possible to even construct an undercutter $c = \langle \{pri, min \rightarrow \neg pri\}, \neg min \rangle$ for the argument b . The agent has then to pay twice the cost of an Argue move. Note, however that from $\text{PROP}(A_i)$ it is not possible to construct an argument whose conclusion is $\neg pub$. This means that the agent is still coherent w.r.t. its assertion ($\neg pub$). Thus, there is no cost to pay for the assert move.

6 Related work

The first standard agent communication languages are KQML [3] and FIPA-ACL [4]. Both languages have been given a mentalistic semantics. The semantics is based on a notion of speech act close to the concept of illocutionary act as developed in speech act theory [9]. Such semantics assumes, more or less explicitly, some underlying hypothesis in particular, that agents are sincere and cooperative. While this may be well fitted for some special cases of interactions, it is obvious that negotiation dialogues are not cooperative. Another more important limitation of this approach is the fact that it is not verifiable since it is based on agents mental states. Our semantics does not refer at all to the mental states of the agents. Moreover, it treats another speech act, namely **Argue**, which allows agents to exchange arguments.

In the second approach, called *social* and developed in [2, 10], primacy is given to interactions among agents. The semantics is based on social *commitments* brought about by performing a speech act. For example, by affirming a data, an agent commits on the truth of that data. After a promise, the agent is committed carrying it out. There are several weaknesses of this approach and we summarize them in the three following points: 1) The definition of commitments complicates the agent architecture in the sense that it needs an ad hoc apparatus. Commitments are introduced especially for modeling communication. Thus agents should reason not only on their beliefs, etc, but also on commitments. In our approach, we didn't introduce any new language to treat commitments. 2) The level at which communication is treated is very abstract, and there is a considerable gap to fill in order to bring the model down to the level of implementation. However, the semantics presented in this paper can be implemented easily. 3) The concept of commitment is ambiguous and its semantics is not clear. According to the speech act, the semantics of the commitment differs. For example, by affirming a data, an agent commits on the truth of that data. The meaning of the commitment here is not clear. It may be that the agent can justify the data or can defend it against any attack, or that the agent is sincere. In our approach, the semantics of a commitment is very intuitive, unique and simple. Penalties are computed in a very simple way and at any time during a dialogue.

The last approach developed by Pitt and Mamdani in [8] and Alberti et al. in [1] is based on the notion of protocol. A protocol defines what sequences of moves are conventionally expected in a dialogue. The meaning of a speech act equates to the set of possible following answers. However, protocols are often technically finite state machines. This turns out to be too rigid in several circumstances. Current research aims at defining flexible protocols, which rely more on the *state of the dialogue*, and less on dialogue history. This state of dialogue is captured by the notion of commitment.

7 Conclusion and perspectives

This paper has introduced a new simple and verifiable ACL semantics. The interpretation of each speech act equates to the penalty to be paid in case the commitment induced by that speech act is violated. In this semantics, a violation criterion is given for each considered speech act. Note that in order to add a new speech act, one needs simply to define a new violation criterion associated with it. This semantics is based on propositional logic, and the violation criteria amount to compute arguments.

An extension of this work to first order logic is under study. Another interesting extension would be to handle explicitly time in order to be able to deal with deadlines for instance.

The notion of penalty may play a key role in defining agent's *reputation* and *trust* degrees. It is clear that an agent that pays a lot of penalties during dialogues may lose its credibility, and will no longer be trusted. Examining more deeply penalties can help to figure out agents profiles: cooperative agent, consistent agent, thoughtful agent (i.e., agent which respects its promises)...

Another possible refinement consists of introducing granularity in the definition of the function **Cost**. The basic idea is to take into account the content of moves when defining their costs. This captures the idea that, for instance, some asserted propositions are more important than others. For example, affirming that the weather is beautiful can be less important than affirming that the president is dead. Our semantics satisfies interesting properties that show its well-foundedness. It also offers other advantages regarding dialogue protocols. For instance, one does not need to specify the different moves allowed after each move in the protocol itself. Agents only need to minimize the penalty to pay at the end of the dialogue. This give birth to very flexible protocols.

REFERENCES

- [1] M. Alberti, A. Ciampolini, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni, 'A social ACL semantics by deontic constraints', in *3rd Int. Central and Eastern European Conference on Multi-Agent Systems CEEMAS'2003, volume 2691 of LNAI*, pp. 204–213, (2003).
- [2] M. Colombetti, 'A commitment-based approach to agent speech acts and conversations', in *Proceedings of the Workshop on Agent Languages and Conversation Policies. 14th International Conference on Autonomous Agents*, pp. 21–29, (2000).
- [3] T. Finin, Y. Labrou, and J. Mayfield, 'Kqml as an agent communication language', in *J. Bradshaw, ed. Software agents, MIT Press, Cambridge*, (1995).
- [4] FIPA, 'Agent communication language', in *FIPA 97 Specification, Foundation for Intelligent Physical Agents.*, (1997).
- [5] C. L. Hamblin, *Fallacies*, Methuen, London, UK, 1970.
- [6] J. Lang, P. Liberatore, and P. Marquis, 'Conditional independence in propositional logic', *Artificial Intelligence*, Volume **141**(1-2), 79–121, (2002).
- [7] J. MacKenzie, 'Question-begging in non-cumulative systems', *Journal of philosophical logic*, **8**, 117–133, (1979).
- [8] J. Pitt and A. Mamdani, 'A protocol based semantics for an agent communication language', in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pp. 486–491, (1999).
- [9] J. R. Searle, 'Speech acts', in *Cambridge University Press*, (1969).
- [10] M. P. Singh, 'A social semantics for agent communication languages', in *IJCAI'99 Workshop on Agent Communication Languages*, pp. 75–88, (1999).
- [11] D. N. Walton and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, State University of New York Press, Albany, NY, 1995.
- [12] M. Winslett, *Updating Logical Databases*, Cambridge University Press, Cambridge, UK, 1990.
- [13] M. Wooldridge, 'Reasoning about rational agents', in *Mit Press*, (2000).

Computational Opinions

Felix Fischer¹ and Matthias Nickles²

Abstract. Existing approaches to knowledge representation and reasoning in the context of open systems either deal with “objective” knowledge or with beliefs. In contrast, there has been almost no research on the formal modelling of *opinions*, i.e., communicatively asserted ostensible beliefs. This is highly surprising, since opinions are in fact the only publicly visible kind of knowledge in open systems, and can neither be reduced to objective knowledge nor to beliefs. In this paper, we propose a formal framework for the representation of dynamic, context-dependent and revisable *opinions* and *ostensible intentions* as a sound basis for the external description of agents as obtained from observable communication processes. Potential applications include a natural semantics of communicative acts exchanged between truly autonomous agents, and a fine-grained, statement-level concept of trust.

1 INTRODUCTION

In the area of artificial intelligence, open systems with heterogeneous, more or less inscrutable actors, are becoming an increasingly important area of application for multi-agency. Prominent examples include the Semantic Web or Peer-to-Peer systems. Nevertheless, agents (including human users) are still mainly characterised in terms of their mental attitudes. The problem with this approach is that agents’ internal states are, by definition, concealed from the point of view of external observers, including other agents. And even if the mental states of all participants were accessible, it would be an extremely difficult task to describe the dynamic behaviour of a system as a function of all of its individuals, particularly if the number of actors is large (as it is usually the case in Web and Peer-to-Peer applications). Hence, although the description of an agent in terms of his beliefs, intentions and desires is a very natural and powerful tool, it reaches its limits rather quickly in complex open environments.

One way to cope with this situation, in order to ensure the controllability of individual agents as well as the achievement of supra-individual system goals, is to impose normative restrictions on agent behaviour, or to model the entire system in terms of static organisational structures. While such top-down approaches certainly have their merits, and systems without predefined policies and protocols are hardly imaginable, they may seriously limit the desirable properties of autonomous systems, like their robustness and flexibility. For example, the use of an agent communication language with a predefined mentalistic semantics [see, e.g., 4] may reduce the level of agent autonomy by prescribing to some degree what the communication parties have to “think” during communication. This is not to dismiss social structures and norms, of course. Public communication models like an ACL semantics cannot avoid normativity if they are to ensure that meaningful communication can take place. We

think, however, that normativity should be kept to a minimum, and the focus should be on adaptive models and on the measurement of the degree to which agents adhere to given models at run-time. After all, agent technology is basically a bottom-up approach, allowing for the emergence of behaviour and solutions, and should not be turned upside down.

Another approach, somewhat complementary to the ones just described, is to constrain the modelling of black-box agents from an external point of view by limiting the *validity* of cognitive models or commitments, thus inducing some sort of bounded observer rationality. A particular way of doing this is by using information about the trustability and reputation of the respective agent. The approach introduced in [13] and refined in this paper is largely compatible and in line with such means, but takes a different perspective. Instead of adding restrictions to statements about internal agent properties and commitments, we introduce the *communication attitudes* (CAs) *opinion* (also called *ostensible belief*) and *ostensible intention* as communication-level counterparts to belief and intention, and thus “lift” mental attitudes to the social level. As for the intended effect, this is similar to approaches using *social commitments* [18]. However, the latter have not yet been fully formalised, and there currently exists no consensus about the precise meaning of “being committed”. In contrast, CAs will be given an intuitive but precise formal semantics which resembles the modalities of agent belief and intention in many ways, and can be used as a direct replacement for belief and intention in most imaginable scenarios (e.g., in the semantics of KQML/KIF or FIPA ACL). CAs are nevertheless cleanly separated from mental attitudes, and can thus be used together with these without any interference. For example, an agent might reason simultaneously about the (alleged) “real” beliefs of another agent and the opinions held communicatively by this agent.

To the best of our knowledge, the formal, systematic treatment of opinions in the above sense, i.e., by distinguishing between rational mental attitudes on the one hand and (boundedly) rational yet “superficial” stances an agent exposes in discourse on the other, is new in the field of AI. This is highly surprising since opinions are in fact the only publicly visible kind of knowledge in open systems, and can neither be reduced to objective knowledge nor to beliefs. Unlike objective knowledge, opinions need not be grounded, shared or consistent. What distinguishes an agents CA of opinion from his mental attitude of belief is that the former is triggered and revised by social conditions, more precisely communicative acts and underlying social structures like legal, organisational and economic laws. While opinions might as well reflect the true beliefs of benevolent, trustworthy agents (an assumption that is made by most traditional approaches to agent communication semantics), this should be considered a special case for autonomous, self-interested agents in open systems. Also, the utterance of an opinion should not mean that one truthfully intends to make someone adopt this opinion as a belief (which would be unrealistic), but rather as an opinion. Opinions emerge from (more or less hidden) agent intentions and social processes, they are tailored to the intended communicative effect and

¹ Computer Science Department, University of Munich, 80538 Munich, Germany, email: fischer@tcs.ifi.lmu.de

² Computer Science Department, Technical University of Munich, 85748 Garching, Germany, email: nickles@in.tum.de

to the opinions and ostensible intentions of the audience. Thus, traditional concepts like epistemic logic, knowledge acquisition, belief ascription and revision, etc., do not necessarily apply to opinions (for example, an opinion could be bought in exchange for money). Yet another approach, which many people use at least implicitly, would be to “abuse” mental attitudes and combine them with additional assumptions like the trustability of agents to model agent interaction. Related but somewhat more appropriate is the ascription of *weak intentions*, which are not necessarily pursued until the intended effect has been achieved or is deemed impossible [2]. This seems workable, since even ostensible intentions and opinions have to built upon some undeniable mental intentions (totally unconscious communication exempted), possibly with a duration of self-commitment that is shorter than claimed. However, it would again lead to problems with cognition and sociality getting mixed up, and force a socially reasoning communication observer to find an immediate, possibly complicated mapping of alleged attitudes to weak intentions. We think that such an amalgamation of mental and pseudo-communication attitudes, while still common, is dangerous and inappropriate. It is likely to impede the simultaneous reasoning about mental and social issues, and it is misleading with respect to the very nature of communication processes and assertions, which form a system of their own that is in a certain sense decoupled from the underlying mental cognition processes [12]. While we will also introduce a notion of trust tailored to the use with opinions, our approach resides on a different conceptual level than related areas like belief revision and information integration. While the latter are mainly concerned with the determination of correct, consistent and useful information, our primary goal is to represent communicatory properties of and differences between heterogeneous assertions, which precedes a potential assessment in terms of reliability.

2 A COMMUNICATION ATTITUDE LOGIC

In this section, we introduce a probabilistic and dynamic *Communication Attitude Logic CAL* as an extension of [7, 2, 1] with additional modalities for opinions and ostensible intentions. Our main goal here is to “lift” the usual modal logic and protocol languages for modelling agent beliefs and intentions and their revision [see, e.g., 8] to the communication (i.e., social) level, in order to deal with mentally opaque agents in open systems. Within the scope of this work, we have chosen a probabilistic variant of dynamic logic as a basis for this, but other formalisms like the *event calculus* [10] could certainly be used instead. Even more importantly, the underlying notions of opinion and ostensible intention do not depend in any way on a dynamic or probabilistic interpretation. As we have pointed out in the previous section, an important property of CAs is their dependency on social conditions, demarcating them, *inter alia*, from mental attitudes. Since an exhaustive description of all the possible ways by which CAs are steered by social conditions of various kinds is beyond the scope of this paper, we confine ourselves to the presentation of simple trigger events.

2.1 Syntax

Definition 1 The language \mathcal{L} of well-formed CAL formulas φ, ψ and processes α, β is given by

$$\begin{aligned} \varphi, \psi ::= & P(V_1, V_2, \dots) \mid \top \mid \perp \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \\ & \mid \varphi \leftrightarrow \psi \mid \diamond \varphi \mid \square \varphi \mid \langle \alpha \rangle p \mid \text{done}(\alpha) \mid \text{happens}(\alpha) \\ & \mid \text{Op}(A_1, A_2, \varphi) \mid \text{OInt}(a_1, A_2, \varphi) \mid \text{Bel}_d(a_1, \varphi) \mid \text{Int}(a_1, \varphi) \end{aligned}$$

$$\alpha, \beta ::= \text{act} \mid a_i.\text{act} \mid \text{any} \mid \alpha; \beta \mid \alpha \cup \beta \mid \alpha^* \mid \varphi?$$

where

- $a_i \in A$ and $A_i \subseteq A$ denote an agent or a set of agents, the source or addressee of an opinion;
- P is a predicate symbol, V_i are variables;
- $\text{act}, a_i.\text{act}$ is an elementary action, possibly indexed with the acting agent a_i ;
- any is an arbitrary action;
- $\alpha; \beta$ denotes sequential process combination;
- $\alpha \cup \beta$ denotes non-deterministic choice between α and β ;
- α^* denotes zero or more iterations of α ;
- $\varphi?$ is a test action (i.e., the process proceeds if φ holds true);
- $\langle \alpha \rangle p$ denotes that α is processed and p holds afterwards;
- $\text{Op}(A_1, A_2, \varphi)$ denotes that agents A_1 hold opinion (ostensible belief) φ facing agents A_2 .
- $\text{OInt}(a_1, A_2, \varphi)$ denotes that agent a_1 , facing agents A_2 , exhibits the ostensible intention to make φ become true (either by himself or indirectly via agents A_2);
- $\text{Bel}_d(A_1, \varphi)$ denotes that every agent $a_1 \in A_1$ (sincerely) believes φ with degree d ; and
- $\text{Int}(a_1, \varphi)$ denotes that a_1 (sincerely) intends φ .

$P(V_1, V_2, \dots), \neg, \wedge, \top, \perp, \rightarrow$ and \leftrightarrow shall have the usual meaning. The meaning of $\text{done}(\alpha)$, $\text{happens}(\alpha)$, $\diamond \varphi$, and $\square \varphi$ is that α has just happened, or is currently taking place, and that φ will hold eventually or always, respectively. The set A may include both agent and non-agent opinion resources (like a Web document). For the sake of readability, we will henceforth refer to all of these resources as *agents*. We further use the following abbreviations:

$$\begin{aligned} \text{Op}(a_1, A_2, \varphi) &\equiv_{\text{def}} \text{Op}(\{a_1\}, A_2, \varphi) \\ \text{Op}(A_1, a_2, \varphi) &\equiv_{\text{def}} \text{Op}(A_1, \{a_2\}) \\ \text{OInt}(a_1, a_2, \varphi) &\equiv_{\text{def}} \text{OInt}(a_1, \{a_2\}, \varphi) \\ \text{Bel}_d(a_1, \varphi) &\equiv_{\text{def}} \text{Bel}_d(\{a_1\}, \varphi) \end{aligned}$$

As an example, consider the sequence of negotiation dialogues given in Table 1. An agent a_2 is offering some item o for sale, and agents a_1 and a_3 are both interested in buying this item. Agents a_1 and a_2 shall be mentally opaque from our point of view, but we shall know the beliefs of agent a_3 (e.g., we could ourselves be a_3). First of all, agent a_3 tells agent a_2 that he would be willing to pay \$100 for o (Step 1a). At the same time, he tells a_2 that a_1 is notoriously unreliable, while privately believing the opposite to be the case (Step 1b). Then, a_1 enters the scene, and a_3 tells a_2 (with a_1 observing) that a_1 is a reliable customer (Step 1c). In a separate dialogue, agent a_1 tells a_2 that he wants to buy item o for \$150 (Step 2a). Agent a_2 refuses this offer, insincerely telling a_1 that agent a_3 would be willing to pay \$200 for o (Step 2b). a_1 then asks a_3 whether he is actually willing to pay \$200, as claimed by a_2 (Step 3). This is when he becomes aware that a_2 (seemingly) tried to cheat him, and tries to extort a lower selling price by telling a_2 that he would otherwise reveal the attempted cheat to a_1 (Step 4). This threat is again insincere, as in fact a_3 does not plan to expose a_2 should he refuse the new offer.

The relevant internal states of a_1 and a_2 being invisible, this scenario cannot be modelled in terms of mental attitudes (e.g., within the BDI framework). This not only applies to the obvious cases of Steps 1b and 4, where $\text{Bel}(a_3, \text{reliable}(a_1))$ and $\text{Op}(a_3, a_2, \neg \text{reliable}(a_1))$, or $\text{OInt}(a_3, a_2, \text{buyFor}50)$ and $\neg \text{Int}(a_3, \text{buyFor}50)$, hold simultaneously. It is tempting to write $\text{Int}(a_2, \text{Bel}(a_1, \text{Int}(a_3, \text{buyFor}200)))$ to model the state after Step 2b,

$\langle a_3.request(buyFor100, a_2) \rangle OInt(a_3, a_2, buyFor100)$	(1a)
$Bel(a_3, reliable(a_1)) \wedge \langle a_3.inform(\neg reliable(a_1), a_2) \rangle$	(1b)
$Op(a_3, a_2, \neg reliable(a_1))$	
$\langle a_3.inform(reliable(a_1), \{a_2, a_1\}) \rangle Op(a_3, \{a_2, a_1\}, reliable(a_1))$	(1c)
$\langle a_1.request(buyFor150, a_2) \rangle OInt(a_1, a_2, buyFor150)$	(2a)
$\langle a_2.inform(OInt(a_3, a_2, buyFor200), a_1) \rangle$	(2b)
$Op(a_2, a_1, OInt(a_3, a_2, buyFor200), a_1))$	
$\langle a_1.request(confirm(OInt(a_3, a_2, buyFor200)), a_3) \rangle$	(3)
$Bel(a_3, Op(a_2, a_1, OInt(a_3, a_2, buyFor200)))$	
$\langle a_3.request(buyFor50, a_2); a_3.inform(done(a_2.refuse)?;$	(4)
$a_3.inform(\neg OInt(a_3, a_2, buyFor200), a_1) \rangle$	
$OInt(a_3, a_2, buyFor50) \wedge OInt(a_3, a_2, done(done(a_2.refuse)?;$	
$a_3.inform(\neg OInt(a_3, a_2, buyFor200), a_1)) \rangle$	
$\wedge \neg Int(a_3, done(a_3.inform(\neg OInt(a_3, a_2, buyFor200), a_1))) \rangle$	

Table 1. CAL formalisation of a sequence of dialogues, involving speech acts *inform*, *request*, *confirm*, and *refuse*.

but this would be mere speculation. For example, a_2 may just have tried to provoke the reaction of Step 4 and then denounce a_3 as an extortionist. The example further highlights the ability of ostensible attitudes to model mutual inconsistencies between information circulating in a group of agents and in one of its subgroups, where after Steps 1b and 1c we simultaneously have $Bel(a_3, reliable(a_1))$, $Op(a_3, a_2, \neg reliable(a_1))$, and $Op(a_3, \{a_2, a_1\}, reliable(a_1))$.

2.2 Semantics

The model theory of \mathcal{L} is that of a first-order linear time temporal logic with modal operators for gradual belief and for intention. For belief, we use (a linear time extension of) the probabilistic semantics proposed in [1], which has been shown to include a KD45 modal operator for full belief as a special case. Intention is modelled by a KD modal operator. We further include two modal operators for the CAs of opinion and ostensible intention. Their individual Kripke-style semantics is roughly equivalent to that of beliefs or intentions, a possible axiomatisation will be discussed in more detail in Section 2.3.

Definition 2 A model is a tuple $M = (\Theta, E, W, A, (\mu_a)_{a \in A}, (I_a)_{a \in A}, (O_{A_1, A_2})_{A_1, A_2 \subseteq A}, (J_{a_1, A_2})_{a_1 \in A, A_2 \subseteq A}, \Phi)$ where Θ is a universe of discourse, E is a set of primitive event types, $W \subseteq \{w : [n] \mapsto E \mid n \in \mathbb{N}\}$ is a set of possible courses of events (or worlds) specified as a total function from the non-negative integers up to n to elements of E . A is a set of agents, and for all $a \in A$, $\mu_a : S \mapsto [0, 1]$ is a discrete probability distribution over situations (i.e., worlds at a particular time step, defined as $S = \{(w, i) \mid w : [n] \mapsto E, n \in \mathbb{N}, 1 \leq i \leq n\}$) for gradual belief, and $I_a \subseteq S \times W$ is a serial accessibility relation for intentions. For all $A_1, A_2 \subseteq A$, $a_1 \in A$, $O_{A_1, A_2} \subseteq S \times W$ is a serial, transitive, Euclidean accessibility relation for opinions, and $J_{a_1, A_2} \subseteq S \times W$ is a serial accessibility relation for ostensible intentions. Φ interprets predicates.

Since S contains at most a denumerable number of situations, μ can be naturally extended to a probability distribution over subsets $T \subseteq S$ by $\mu(T) = \sum_{s \in T} \mu(s)$. In particular, we have $\mu(S) = 1$.

Formulas of the logical language \mathcal{L} are interpreted with respect to a tuple (M, v, σ, i) , consisting of a model M , a variable assignment v (mapping variables to elements of Θ), a possible world (i.e., possible course of events) σ , and a time step i in this particular world. The interpretation of formulas is given by the standard interpretation rules

for first-order formulas and for the action-related modal operators *happens* and *done*. We refer to [2] for formal definitions of these rules. The gradual belief operator Bel_p , the intention operator Int , the opinion operator Op , and the ostensible intention operator $OInt$ are interpreted as follows:

$(M, v, \sigma, i) \models Bel_p(a, \varphi)$	iff $\mu_a(\{s' \mid (M, v, s', i) \models \varphi\}) = p$
$(M, v, \sigma, i) \models Int(a, \varphi)$	iff for all σ' such that $(\sigma, i, \sigma') \in I_a$, $(M, v, \sigma', n) \models \varphi$
$(M, v, \sigma, i) \models Op(A_1, A_2, \varphi)$	iff for all σ' such that $(\sigma, i, \sigma') \in O_{A_1, A_2}$, $(M, v, \sigma', n) \models \varphi$
$(M, v, \sigma, i) \models OInt(a_1, A_2, \varphi)$	iff for all σ' such that $(\sigma, i, \sigma') \in J_{a_1, A_2}$, $(M, v, \sigma', n) \models \varphi$

That is, φ is *gradually believed* by agent a with strength p iff the measure μ_a of all situations where φ holds true is at least p , and *intended* by a iff it holds in all situations that are accessible via I_a . φ is *ostensibly believed* (*ostensibly intended*) by A_1 facing A_2 (a_1 facing A_2) iff it holds in all situations that are accessible via O_{A_1, A_2} (J_{a_1, A_2}). While an agent would usually relate the strength of belief to past events (i.e., frequency) in some way, we will not commit to a particular way this is done in the context of this paper (e.g., by giving axioms that relate μ to past events).

2.3 Communication Attitudes

Even though agents are fully autonomous, CAs should be governed by a set of axioms that separate rational and irrational communicative behaviour. As we have already outlined above, violation of these axioms would, in the most extreme case, disqualify an agent from taking part in (rational) interaction with other agents. The following axioms describe (the consistency of a set of) CAs in a way that is similar to a KD45 logic of belief and a KD logic of intention, but cleanly separated from these mental attitudes:

1. $Op(A_1, A_2, \varphi \rightarrow \psi) \rightarrow (Op(A_1, A_2, \varphi) \rightarrow Op(A_1, A_2, \psi))$
2. $Op(A_1, A_2, \varphi) \rightarrow \neg Op(A_1, A_2, \neg \varphi)$
3. $Op(A_1, A_2, \varphi) \rightarrow Op(A_1, A_2, Op(A_1, A_2, \varphi))$
4. $\neg Op(A_1, A_2, \varphi) \rightarrow Op(A_1, A_2, \neg Op(A_1, A_2, \varphi))$
5. $OInt(a_1, A_2, \varphi \rightarrow \psi) \rightarrow (OInt(a_1, A_2, \varphi) \rightarrow OInt(a_1, A_2, \psi))$
6. $OInt(a_1, A_2, \varphi) \rightarrow \neg OInt(a_1, A_2, \neg \varphi)$
7. $OInt(a_1, A_2, \varphi) \leftrightarrow Op(a_1, A_2, OInt(a_1, A_2, \varphi))$
8. $OInt(a_1, A_2, \varphi) \leftrightarrow Op(A_2, a_1, OInt(a_1, A_2, \varphi))$
9. $\neg OInt(a_1, A_2, \varphi) \leftrightarrow Op(a_1, A_2, \neg OInt(a_1, A_2, \varphi))$
10. $\neg OInt(a_1, A_2, \varphi) \leftrightarrow Op(A_2, a_1, \neg OInt(a_1, A_2, \varphi))$
11. $OInt(a_1, A_2, \varphi) \rightarrow \neg Op(a_1, A_2, \varphi)$
12. $OInt(a_1, A_2, Op(A_2, a_1, \varphi)) \rightarrow Op(a_1, A_2, \varphi)$
13. $Op(a_1, A_2, \varphi) \rightarrow \langle A_2.query(a_1, \varphi); a_1.reply(A_2, \varphi) \rangle \top$

Axioms 1 to 6 correspond to the usual axioms of a KD45 modal operator of belief and to that of a KD operator of intention, and are additionally contextualised with both sender and addressee. The remaining axioms concern additional properties of opinions, ostensible intentions, and the relationship between them. Axiom 13 states that agents (have to) admit their opinions at least upon request, if this request is observed by the opinion addressees. In practice, the above axioms, which provide a basic, abstract set of social norms, could be augmented by additional rules concerning “etiquette” in social interaction and emergent social norms (e.g., under what circumstances it is socially acceptable to retract an assertion or drop an intention).

For opinions that are not just held and uttered on request (like in an opinion poll), but actively asserted facing other agents, we can

define a special kind of *active opinion*, denoted Op^a . Since the latter includes the ostensible intention to spread the opinion among the respective addressees, its semantics can directly be given in terms of ostensible intentions, i.e.,

$$Op^a(a_1, A_2, \varphi) \equiv_{\text{def}} OInt(a_1, A_2, Op(A_2, a_1, \varphi)).$$

By Axiom 12, we have $Op^a(a_1, A_2, \varphi) \rightarrow Op(a_1, A_2, \varphi)$, i.e., *passive opinions* form the consequential closure of those pro-actively put forward in terms of communication. By Axioms 8 and 12, we further have $Op^a(a_1, A_2, \varphi) \rightarrow Op(A_2, a_1, Op^a(a_1, A_2, \varphi))$, a variation of Axiom 3 for the addressee. We do not distinguish between active and passive ostensible intentions, because in this case consequential closure mainly concerns functional decomposition and side-effects of actions, of which an agent should be aware if he has actively announced them.

We further propose the following axioms to bridge the gap between mental and communication attitudes. In most cases, however, these will not be needed to reason about communication processes.

14. $OInt(a_1, A_2, \varphi) \leftrightarrow Bel(a_1, OInt(a_1, A_2, \varphi))$
15. $OInt(a_1, A_2, \varphi) \leftrightarrow Bel(A_2, OInt(a_1, A_2, \varphi))$
16. $Op(A_1, A_2, \varphi) \leftrightarrow Bel(A_1, Op(A_1, A_2, \varphi))$
17. $Op(A_1, A_2, \varphi) \leftarrow Bel(A_2, Op(A_1, A_2, \varphi))$
18. $Op(a_1, a_1, \varphi) \leftrightarrow Bel(a_1, \varphi)$
19. $OInt(a_1, a_1, \varphi) \leftrightarrow Int(a_1, \varphi)$

According to Axiom 17, (passive) opinions don't necessarily have to be visible, but if we come to believe that someone holds an opinion, this belief is always justified. Axioms 18 and 19 allow us to write (full) belief and intention as special cases of opinion and ostensible intention, respectively.

2.4 Communication Attitudes and Communication

Having separated CAs from mental attitudes, the natural next step is to interpret communication in terms of CAs. In the context of this paper, we do not aim at an axiomatisation of a comprehensive set of speech acts like that of FIPA ACL [4], hence giving them an alternative social (but not commitment-based) semantics in terms of CAs. Instead, we provide a minimal set of four communicative acts, and relate CAs directly to actual communications using these communicative acts. *inform* asserts that a particular logical formula currently holds true, *request* declares the intention to make a formula true (note that in our framework, intentionality may include bringing about a certain state or action indirectly “using” other agents), *retract* and *dismiss* cancel a previous statement. Communicative acts are denoted $sender.act(content, addressee)$.

The following *trigger axioms* describe the functional relationship between these communicative acts on the one hand and opinions and ostensible intentions on the other (we use a binary *weak until* operator which can be defined as $\psi W \psi' \equiv_{\text{def}} \text{happens}((\psi?)^*; \psi') \vee \square \psi$):

20. $(Op) done(a_1.inform(\varphi, A_2)) \rightarrow (Op(a_1, A_2, \varphi) W done(a_1.retract(\varphi, A_2)))$
21. $(OInt) done(a_1.request(\varphi, A_2)) \rightarrow (OInt(a_1, A_2, \varphi) W done(a_1.dismiss(\varphi, A_2)))$

As we have already said, additional rules will be necessary to restrict the use of *retract* and *dismiss* to situations in which it is socially acceptable to retract an assertion or drop an intention. Again, such rules are not to prevent an autonomous agent from uttering certain communication primitives in a given context, but rather define what is considered rational communication and what is not.

3 TRUSTABILITY

Kripke-style semantics are widely used to model rational attitudes [see, e.g., 19], and we have pointed out that we consider CAs rational at least to some degree. Yet, they are not very helpful when it comes to determining the actual consequences of someone holding a certain CA (e.g., in terms of reliability). This is crucial because communication is, from an observers point of view, much about decision, behavioural expectation and prediction. As an example for such a *consequentialist semantics* of opinions and ostensible intentions, one might expect that holding a certain ostensible intention means to act “in accordance” with this attitude at least for a limited amount of time, i.e.,

$$OInt(a_1, A_2, done(\alpha_1; \dots; \alpha_n)) \rightarrow \exists h \geq 0. Int(a_1, done(\alpha_1; \dots; \alpha_h))$$

That is, an ostensible intention regarding a sequence of actions implies that the agent *truly* intends a (possibly empty) prefix of this sequence. A similar assumption could be made for opinions. As a matter of rationality in communication, a prefix should truly be intended if it needs to be performed by a_1 alone and under observation by A_2 (i.e., $h \geq j$ instead of $h \geq 0$ if $\alpha_i = a_1.\alpha_i$ and is observable by A_2 for $i \leq j$). Here, h can be seen as the *sphere of trustability* of the respective CA, i.e., the length of the largest sequence of actions that are truly intended. Due to the mental opacity of a_1 , the value of h would in practice have to be determined in a context-dependent way via empirical observation of past ostensible intentions and their consequences in terms of actions actually executed by a_1 . Such a mapping of ostensible to mental attitudes would already provide a consequentialist semantics for ostensible intentions and beliefs, but would come at the price of grounding CAs in mental attitudes. Moreover, it is not at all clear whether and how the extent of the sphere could be computed from past observations, given that intentions denote an earnest self-commitment [17]. To reflect this insight and still allow an observer of a CA to derive information regarding static and dynamic aspects of the world, we propose a consequentialist semantics of CAs that is based on the observer's *trust* that a particular CA does indeed describe (part of) the worlds current state and future evolution.

Trust is commonly defined as a belief that another party will do what it says it will, i.e., be honest and reliable, or reciprocate, i.e., return an advance action for the common good of both, given an opportunity to defect to get higher payoff [see, e.g., 16]. Hence, trust provides an agent a_1 with a means to derive expectations about a certain issue (proposition) φ from CAs regarding φ directed toward him by other agents A_2 . This kind of trust may become crucial when a_1 himself is uncertain about φ , but forced to take impromptu action depending on φ . We give a formal definition.

Definition 3 Let a_1 be an agent, A_2 a set of agents, φ a proposition. Then trusting with regard to φ and with degree p in light of a necessary decision is defined as

$$\begin{aligned} DTrust_p(a_1, A_2, \varphi) &\equiv_{\text{def}} \\ &(happens(a_1.(\alpha \cup \beta)) \\ &\wedge Int(a_1, done(Bel(a_1, \varphi)?; a_1.\alpha \cup Bel(a_1, \neg\varphi)?; a_1.\beta))) \\ &\wedge Bel_q(a_1, \varphi) \wedge (q < 1) \wedge (p + q = 1) \\ &\wedge Op(A_2, a_1, \varphi)) \leftrightarrow Int(a_1, done(a_1.\alpha)). \end{aligned}$$

That is, if (i) agent a_1 is forced to choose between actions α and β and intends to do so depending on whether or not φ holds, (ii) a_1 believes in φ with degree $q < 1$, and (iii) some other agents A_2 hold the opinion against a_1 that φ , then a_1 is said to trust A_2 with respect

to φ and with degree p if he takes action α if and only if $p + q = 1$. A similar definition could be given for ostensible intentions.

Current research on trust in the area of artificial agents mainly focuses on *trust metrics*, which determine how trust and reputation of different dimensions and from various sources are aggregated to form a single notion of trust [see, e.g., 9, 6]. A major weakness of these purely numerical approaches, especially when a precise justification or motivation is required, is the lack of a formal semantics. Existing logical models of trust, on the other hand, including those used in the context of belief revision to assess new information, ground these semantics in mental attitudes of the participating agents, particularly those of the *trustee* [see, e.g., 11, 3]. Like a mentalistic agent communication semantics, this is likely to impede any process by which trust is inferred and used in open environments where agents are mentally opaque. Definition 3 further differs from most existing approaches in that it considers trust at the level of contextualised behavioural expectations or (communicative) assertions rather than entire agents, which results in a more fine-grained and precise model. Since communication is the primary means of interaction among intelligent, autonomous agents, communicative events should constitute the natural, basic subject of trust among such agents. This is in line with the prevailing sociological view of trust as confidence in ones expectations, providing a solution to the problem of acting based on contextualised and gradual (i.e., possibly uncertain) expectations and reducing the inherent complexity of social interaction [12].

While describing precisely (i) what actions need to be taken by an agent who trusts a CA, and (ii) how trust can be recognised ex post, Definition 3 does not provide any additional insights regarding when a CA should or should not be trusted. This underlines the observation of [11] that trust cannot fully be defined as a derived concept. The only available information regarding a decision to trust are past interactions that have taken place in a similar social context or, more precisely, behavioural expectations derived from these interactions. Procedures of arbitrary complexity and sophistication could be imagined to derive trust and hence decisions regarding further action from such expectations. For example, one might want to infer trust (of degree p) in agents A_2 regarding proposition φ from gradual belief (of the same degree) that an opinion held by A_2 that φ does indeed mean φ , i.e., $Bel_p(a_1, Op(A_2, a_1, \varphi) \rightarrow \varphi) \Rightarrow DTrust_p(a_1, A_2, \varphi)$. The left hand side of this rule resembles a special case of *adaptive expectations*, i.e., expectations that are revised in case of a disappointment [15]. A particular algorithm by which adaptive expectations can be learned and updated from actual interactions is given in [14]. For reasons of limited space, we refer to the original publications for details. In any case, this process might involve dealing incompatible beliefs, and will hence have to be combined with appropriate mechanisms for belief update or belief revision.

4 CONCLUSION

Information in open environments emerges from a potentially large number of competing goals and viewpoints and can be both unreliable and inconsistent. Moreover, a commonly agreed “truth”, or the most basic laws governing reasonable ways to come to an agreement, might not exist. Moving towards a semantically rich formalisation of *opinions* must therefore be a core concern of a strictly communication-oriented paradigm of knowledge representation and distributed AI. This has been the motivation underlying the work described here. Future research will have to concentrate on the application of our approach in real-world domains, and on the integration with the closely related approach of *grounding* in discourse and dialogue games [5].

ACKNOWLEDGEMENTS

We would like to thank Andreas Herzig for valuable discussions on the topic of this paper and the anonymous reviewers for helpful comments. Part of this material is based upon work supported by the Deutsche Forschungsgemeinschaft under grant BR 609/13-1.

REFERENCES

- [1] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge*, The MIT Press, 1990.
- [2] P. R. Cohen and H. J. Levesque, ‘Intention is choice with commitment’, *Artificial Intelligence*, **42**, 213–261, (1990).
- [3] R. Demolombe, ‘Reasoning about trust: a formal logical framework’, in *Proc. of the 2nd Int. Conf. on Trust Management*, (2004).
- [4] FIPA. FIPA communicative act library specification, 2002.
- [5] B. Gaudou, A. Herzig, and D. Longin, ‘Grounding and the expression of belief’, in *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, (2006).
- [6] J. Golbeck and J. Hendler, ‘Accuracy of metrics for inferring trust and reputation’, in *Proc. of the 14th Int. Conf. on Knowledge Engineering and Knowledge Management*, (2004).
- [7] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*, The MIT Press, 2000.
- [8] A. Herzig and D. Longin, ‘A logic of intention with cooperation principles and with assertive speech acts as communication primitives’, in *Proc. of the 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*. ACM Press, (2002).
- [9] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, ‘An integrated trust and reputation model for open multi-agent systems’, *Autonomous Agents and Multi-Agent Systems*, (2006). To appear.
- [10] R. A. Kowalski and M. J. Sergot, ‘A logic-based calculus of events’, *New Generation Computing*, **4**(1), 67–96, (1986).
- [11] C.-J. Liau, ‘Belief, information acquisition, and trust in multi agent systems – a modal logic formulation’, *Artificial Intelligence*, **149**(1), 31–60, (2003).
- [12] N. Luhmann, *Social Systems*, Stanford University Press, Palo Alto, CA, 1995.
- [13] M. Nickles, F. Fischer, and G. Weiss, ‘Communication attitudes: A formal approach to ostensible intentions and individual and group opinions’, in *Proc. of the 3rd Int. Workshop on Logic and Communication in Multi-Agent Systems (LCMAS)*, (2005).
- [14] M. Nickles, M. Rovatsos, and G. Weiss, ‘Empirical-rational semantics of agent communication’, in *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AA-MAS)*, (2004).
- [15] M. Nickles, M. Rovatsos, and G. Weiss, ‘Expectation-Oriented Modeling’, *Engineering Applications of Artificial Intelligence*, **18**(8), (2005).
- [16] S. D. Ramchurn, D. Huynh, and N. R. Jennings, ‘Trust in multi-agent systems’, *The Knowledge Engineering Review*, **19**(1), 1–25, (2004).
- [17] Y. Shoham, ‘Agent-oriented programming’, *Artificial Intelligence*, **60**(1), 51–92, (1993).
- [18] Munindar P. Singh, ‘A social semantics for agent communication languages’, in *Proc. of the IJCAI Workshop on Agent Communication Languages*, (2000).
- [19] Michael Wooldridge, *Reasoning about Rational Agents*, The MIT Press, 2000.

A New Semantics for the FIPA Agent Communication Language based on Social Attitudes

Benoit Gaudou¹ and Andreas Herzig¹ and Dominique Longin¹ and Matthias Nickles²

Abstract. One of the most important aspects of the research on agent interaction is the definition of *agent communication languages* (ACLs), and the specification of a proper formal semantics of such languages is a crucial prerequisite for the usefulness and acceptance of artificial agency. Nevertheless, those ACLs which are still mostly used, especially the standard FIPA-ACL, have a communication act semantics in terms of the participating agents' *mental attitudes* (viz. beliefs and intentions), which are in general undeterminable from an external point of view due to agent autonomy. In contrast, semantics of ACLs based on *commitments* are fully verifiable, but not sufficiently formalized and understood yet. In order to overcome this situation, we propose a FIPA-ACL semantics which is fully verifiable, fully formalized, lean and easily applicable. It is based on *social attitudes* represented using a *logic of grounding* in straightforward extension of the BDI agent model.

1 Introduction

The design of agent communication languages (ACLs) has attracted a lot of attention during the last years. Such languages are mostly based on Searle and Vanderveken's speech act theory [9], and are not only relevant for applications involving real software agents or robots, but also for other software entities which need to communicate, like web services.

Among the different existing ACLs, FIPA-ACL is still the most important standard, subsets of which are widely used in agent interaction protocols. FIPA-ACL is semantically rich, and the concepts involved are quite intuitive.

Nevertheless, FIPA-ACL has a feature that has often been criticized in the literature, viz. that the semantics of communication acts (CAs) is defined in terms of the agents' mental states. For example, when agent i informs agent j that φ , then the (rational) effect is that agent j starts to believe φ . In order for such an effect to obtain some hypotheses have to be made; but even in such contexts j is autonomous and might not adopt φ , and in any case i or other agents and the system designer can never verify whether this is the case or not. This is especially felt as being too strong in open environments with black- or gray-box agents where we don't even want to ascribe mental attitudes to other agents.

In contrast, those semantics based on the concept of social commitments [10] is verifiable because they are only based on what has been communicated and the commitments the agents have made by doing that (instead of the beliefs and intentions that are "behind"

these commitments and that have caused them). The drawback here is that the existing approaches are only semi-formal, in particular because there is no consensus on what "being committed" actually means. As a consequence, they are rarely used in practice up to now.

The aim of this paper is to resolve the problems of FIPA's CA semantics without loosing its benefits. We propose a novel semantics avoiding the strong hypotheses of the original semantics by "lifting" the BDI-based FIPA semantics to the social level. We do so by replacing the usual private mental attitudes of BDI logics by *public* mental attitudes, i.e. attitudes that have been made public through communication (*social attitudes*). More precisely, our semantics is based on an unified and extended approach to the concept of *communication attitudes (ostensible beliefs and intentions)* [7] and the more or less equivalent concept of *grounding*³ [5]. For example, the effect of an informing-that- p act is that it is public that the sender believes that p . This does not mean that the sender really believes that p , but only hinders him to subsequently inform that $\neg p$, or to inform that he ignores whether p .

The major benefits of our new semantics are the following:

- It is verifiable, and suitable even for truly autonomous, possibly malevolent agents.
- It is fully formalized.
- It is based on a straightforward extension of BDI, and therefore relatively lightweight.
- It can easily be adapted to similar ACLs, e.g. the widely used KQML/KIF.
- It generalizes the single-addressee FIPA acts to groups of agents, and it distinguishes the group of addressees from the group of bystanders (overhears), and thus refines FIPA's acts.

All in all, we aim at an agent communication semantics that eliminates the major shortcomings of the still predominant mentalist approaches to ACL semantics while being "upward compatible" to the standard FIPA semantics and similar approaches, lean, and formally well founded.

The remainder of this paper is organized as follows: The next section provides a short account of the logical framework that we have chosen as a formal foundation of our approach. Section 3 presents the new semantics, and Section 4 illustrates our approach by means of a case study. Section 5 concludes.

2 A logic of grounding

In this section we briefly present the logic of belief, intention, action, and grounding defined in [5], that is based on Cohen and

¹ Université Paul Sabatier, IRIT, Toulouse, email: gaudou.herzig.longin@irit.fr

² Technical University of Munich, email: nickles@informatik.tu-muenchen.de

³ We use the term *grounding* as Traum [11], i.e. it refers to "the process of adding to the common ground between conversational agent".

Levesque's [2]. AGT is a finite set of agents, ACT is a set of actions, $ATM = \{p, q, \dots\}$ is the set of atomic formulas. Complex formulas are denoted by φ, ψ, \dots . A model \mathcal{M} is a 5-tuple that is made up of: a set of possible worlds W ; a mapping

$$\mathcal{V} : W \longrightarrow (ATM \longrightarrow \{0, 1\})$$

associating a valuation \mathcal{V}_w to every $w \in W$; a mapping

$$\mathcal{A} : ACT \longrightarrow (W \longrightarrow 2^W)$$

associating actions $\alpha \in ACT$ and worlds $w \in W$ with the set of worlds resulting from the execution of α in w ; a mapping

$$\mathcal{G} : (2^{AGT} \setminus \emptyset) \longrightarrow (W \longrightarrow 2^W)$$

associating sets of agents $I \subseteq AGT$ and worlds $w \in W$ with the set of worlds that are publicly possible for the group I at w (the worlds that are compatible with what has been uttered in I 's presence); and finally the mapping

$$\mathcal{I} : AGT \longrightarrow (W \longrightarrow 2^{2^W})$$

associating every $i \in AGT$ and world w with the set of propositions (alias sets of worlds) that are intended by i . (The \mathcal{I}_i are neighborhood functions in Chellas' sense [1].)

The logical language contains modal operators of action $After_\alpha$ and $Before_\alpha$, for every $\alpha \in ACT$, modal operators of groundedness G_I for every group I , and modal operators of intention Int_i for every agent $i \in AGT$.

The formula $After_\alpha \varphi$ reads “ φ is true after every execution of the action α ”, and $Before_\alpha \varphi$ reads “ φ is true before every execution of the action α ”. Semantically, $w \models After_\alpha \varphi$ iff $w' \models \varphi$ for each $w' \in \mathcal{A}_\alpha(w)$, and $w \models Before_\alpha \varphi$ iff $w' \models \varphi$ for each w' such that $w \in \mathcal{A}_\alpha(w')$. The logic of $After_\alpha$ and $Before_\alpha$ is the tense logic K^t, i.e. standard normal modal logic K plus the conversion axioms $\varphi \rightarrow Before_\alpha \neg After_\alpha \neg \varphi$ and $\varphi \rightarrow After_\alpha \neg Before_\alpha \neg \varphi$. The abbreviation $Done_\alpha \varphi \stackrel{\text{def}}{=} \neg Before_\alpha \neg \varphi$ reads “ α has just been done before which φ was true”. We note $Done(\alpha) \stackrel{\text{def}}{=} Done_\alpha \top$ for convenience. Moreover, $Before_{\alpha \cup \alpha'} \varphi$ abbreviates $Before_\alpha \varphi \wedge Before_{\alpha'} \varphi$. (Hence $Done(\alpha \cup \alpha')$ stands for $Done(\alpha) \vee Done(\alpha')$.)

$G_I \varphi$ reads “it is grounded for group I that φ is true”, or for short: “ φ is grounded for I ”. When I is a singleton, $G_{\{i\}} \varphi$ means that for agent i , φ is grounded. In this (and only in this) degenerated case ‘public’ grounding is the same as private belief. We write $G_i \varphi$ for $G_{\{i\}} \varphi$. The accessibility relations of grounding operators must satisfy the constraints for the standard normal modal logic KD (seriality), plus the following, for groups I, I' such that $I' \subseteq I$:

- (i) if $u\mathcal{G}_{I'}v$ and $v\mathcal{G}_Iw$ then $u\mathcal{G}_Iw$
- (ii) if $u\mathcal{G}_{I'}v$ and $u\mathcal{G}_Iw$ then $v\mathcal{G}_Iw$
- (iii) if $u\mathcal{G}_Iv$ and $v\mathcal{G}_{I'}w_1$ then there is w_2 such that $u\mathcal{G}_Iw_2$ and $V(w_1) = V(w_2)$
- (iv) $\mathcal{G}_I \subseteq \bigcup_{i \in I} \mathcal{G}_i \circ \mathcal{G}_i$

Constraint (i) stipulates that subgroups are aware of what is grounded in the group: whenever w is a world for which it is grounded for I' that all I -grounded propositions hold in w , then all I -grounded propositions indeed hold in w . This is a kind of *attention* property: each subgroup participating in a conversation is aware of what is grounded in the group. Similarly (ii) expresses that subgroups are aware of what is ungrounded in the group, too. (i) and (ii) correspond to the axioms of strong rationality (SR+) and (SR-):

$$G_I \varphi \rightarrow G_{I'} G_I \varphi \quad (\text{SR+})$$

$$\neg G_I \varphi \rightarrow G_{I'} \neg G_I \varphi \quad (\text{SR-})$$

which express that if a proposition φ is grounded (resp. ungrounded) for a group I then it is grounded for each subgroup that φ is grounded

(resp. ungrounded) for I ⁴.

(iii) stipulates that for every objective proposition grounded for I it is publicly established for I that each subgroup of I is grounded on it (which does not imply that it is grounded for the latter): whenever w is a world for which all propositions grounded for I' are grounded for I , then all those propositions are indeed grounded for I in w . It validates the axiom (WR)

$$G_I \varphi \rightarrow G_I G_{I'} \varphi, \text{ for } \varphi \text{ objective} \quad (\text{WR})$$

which says that if the objective formula φ is grounded for a group K then it is necessarily grounded for K that for each subgroup K' the formula is grounded.⁵ Note that this does not imply that for every subgroup φ is actually grounded, i.e. (WR) does not entail $G_K \varphi \rightarrow G_{K'} \varphi$. In particular, the fact that φ is grounded for group K does not imply that the members of K believe that φ .

(iv) expresses that if it is grounded for a set I that a proposition is established for every agent then it is grounded for I , too. This corresponds to axiom (CG)

$$\left(\bigwedge_{i \in I} G_I G_i \varphi \right) \rightarrow G_I \varphi \quad (\text{CG})$$

which says that if a proposition is established for every agent in I , then it is established for the whole group I . Together, (WR) and (CG) stipulate that for objective φ we have $(\bigwedge_{k \in K} G_K G_k \varphi) \leftrightarrow G_K \varphi$. Note that $G_K \varphi$ does NOT imply $G_k \varphi$ where $k \in K$. Indeed, a proposition can be grounded in a group independently of the private belief of each agent of the group about this proposition: there is thus no sincerity hypothesis.

$Int_i \varphi$ reads “agent i intends that φ be true”. The Int_i are non-normal modal operators which only validate the rule of equivalence: $\frac{\varphi \leftrightarrow \psi}{Int_i \varphi \leftrightarrow Int_i \psi}$. They neither validate $Int_i(\varphi \wedge \varphi') \rightarrow (Int_i \varphi \wedge Int_i \varphi')$ nor $(Int_i \varphi \wedge Int_i \varphi') \rightarrow Int_i(\varphi \wedge \varphi')$.

Intentions and actions are related by the principle of intentional action saying that if α has just been performed by agent i then i had the intention to do so immediately before.

$$Before_{i:\alpha} Int_i Done(i:\alpha) \quad (\text{IA})$$

where $i:\alpha$ denotes that action α is performed by agent i .

To highlight our proposal for the semantics of grounding consider the following example. There are three agents $AGT = \{0, 1, 2\}$. Let agent 0 (privately) believe that 2 sells high-quality products, formally written $G_0 q_2$. Now suppose that in private conversation agent 0 tells 1 that the contrary is the case (for example to trigger some attitude of 1 that benefits 0). The (illocutionary) effect is $G_{\{0, 1\}} G_0 \neg q_2$. Then agent 2 joins in the conversation, and later on 0 informs 1 and 2 that q_2 : The illocutionary effect is $G_{\{0, 1, 2\}} G_0 q_2$. This illustrates that even for nested groups $\{0\} \subset \{0, 1\} \subset \{0, 1, 2\}$, mutually inconsistent states of public group belief might hold simultaneously.

3 Communication act semantics

Following and extending the ACL syntax used in [4], a single communication act (CA) is denoted as $\langle i, \text{ActName}(J, \varphi), K \rangle$, where i

⁴ In particular, we have the modal axioms (4) and (5) for G_I operators as theorems of our logic.

⁵ (WR) concerns only objective formulas, i.e. formula that does not contain any modality. If we applied (WR) to some mental states, we would restrict the agents' autonomy. For example, when an agent performs the speech act $\langle i, \text{Inform}(J, p), K \rangle$, he expresses publicly that he believes p . Thus if agent i expresses: $\langle i, \text{Inform}(J, G_J p), K \rangle$ the formula $G_K G_i G_J p$ holds, and the agents $j \in J$ cannot afterwards express that they believe $\neg p$. If he made this speech acts, the formulae $G_K G_J \neg p$ and, thanks to (WR), $G_K G_i G_J \neg p$ would hold, which is inconsistent with the above formula $G_K G_i G_J p$.

is the performing agent, J is a group of recipients (whereas FIPA only allows one addressee). $ActName$ is the name of the act (in our model not necessarily corresponding to exactly one speech act type, see below). φ is the propositional content of the act. K , which is missing in FIPA, denotes a group of attending agents who overhear the respective utterance, with $i \in K$, $J \subseteq K \setminus \{i\}$ and $J \neq \emptyset$. For a dialogue of only two agents i and j we have $J = \{j\}$ and $K = \{i, j\}$.

In the standard semantics of FIPA CAs [4] (henceforth called FIPA-S), semantics is specified by providing the *feasibility preconditions* (FPs) and the *rational effects* (REs) of single CAs. The former denote which logical conditions need to be fulfilled in order to execute the respective act, and the latter specify which conditions hold after the successful performance of that act. FPs characterize both the ability of the speaker to perform the act and the context-dependent relevance of the act (i.e., that performing the act is relevant given a certain dialogue context). In contrast, REs specify the desired and rationally-expectable direct perlocutionary effect of the utterance, i.e. what becomes true in case the perlocutionary act succeeds.

We think there are at least three reasons not to qualify a CA by its rational effect. Firstly, it is possible to desire and expect different kinds of RE of the same CA; secondly, Searle shows in [9, Sec. 2.1] that the effect of a speech act cannot be a rational (or perlocutionary) effect simply because a lot of speech acts just do not have any perlocutionary effect. He also shows that even if a speech act can have a perlocutionary effect, we can always exhibit a context where the speaker does not intend this perlocutionary effect. Thirdly, strong hypotheses (such as sincerity, competence, credibility...) must be made about the agents to enable the inference of the expected RE, which is too restrictive in our context of open multi-agent systems, possibly with conflicts and malevolent, egocentric agents...

In contrast to FIPA-S, the FPs and IEs (for illocutionary effects) in our model do not make any statement about mental attitudes, but specify the preconditions and effects in terms of groundings of group K (the public, so to say). They are chosen such that the respective communication act is both executable given all realistic preconditions, and succeeds reliably with a publicly verifiable effect. The only (self-evident) exception follows from the bridge axioms (SR_+) and (SR_-) given in the previous section, stating that an agent or subgroup of a certain group knows about the existence of the respective grounded beliefs or intentions of their group — this means merely that the agents keep track of the ongoing course of communication in terms of FPs and IEs.

In the sequel we use the term *Social Attitudes Based Semantics* (SABS) for our modelling, and will define the SABS semantics of the four primitive CAs of FIPA-ACL: *Inform*, *Request*, *Confirm* and *Disconfirm*, and we will also present the respective FIPA-S specifications for comparison. All other FIPA-CAs are macros composed of these primitives in a more or less straightforward manner.

3.1 Inform: Asserting information

We start with the FIPA-S version of the semantics:

$$\langle i, \text{Inform}_{\text{FIPA}}(j, \varphi), K \rangle$$

FP: $\text{Bel}_i \varphi \wedge \neg \text{Bel}_i (\text{Bel}_j \varphi \vee \text{Bel}_j \neg \varphi \vee U_j \varphi \vee U_j \neg \varphi)$

RE: $\text{Bel}_j \varphi$

At this, $U_j \varphi$ denotes that agent j is uncertain about φ , but thinks that φ is more likely than $\neg \varphi$. The terms “uncertain” and “more likely” are not precisely defined in [4]. The essential preconditions of *Inform* in the FIPA-S are thus that agent i truthfully believes what he asserts,

and that the receiver does not have any definite opinion about the asserted proposition. Whereas the former condition is obviously unrealistic given a truly autonomous agent i , the latter disallows (problematically) the use of *Inform* to convince the addressee. We consider the latter usage as crucial e.g. in the context of computational argumentation and argumentation-based negotiation. We could introduce an additional conviction-act extending the syntax, or try to emulate it with a construct like *Request(Inform(φ))*, but this would not only be unnecessary and inelegant, but would also blur the fact that there exists a continual transition from “pure information” to conviction. It is also not clear why the absence of an opinion shall be a realistic precondition for the success of an information act, or, conversely, why the existence of an opinion (which could be very weak, or “by default” only) shall hinder the receiver to adopt the asserted information (e.g., consider that the addressee might trust the sender more than herself).

The rational effect of *Inform* in FIPA-S is simply that the addressed agent believes what she has been told (in case the act succeeds). Of course, this effect cannot be verified with autonomous agents. Even if it could be verified, it would be too strong and unlikely. Moreover it is not verifiable that the informing agent (truthfully) intends the adoption of a certain belief.

These concerns lead to the following SABS semantics:

$$\langle i, \text{Inform}(J, \varphi), K \rangle$$

FP: $\neg G_K G_J \varphi \wedge \neg G_K \text{Int}_i G_J \varphi \wedge \neg G_K \neg G_i \varphi$

IE: $G_K G_i \varphi$

In the FP, $\neg G_K G_J \varphi$ specifies that the addressed agent has not expressed φ before (with group K attending), corresponding to the $\neg \text{Bel}_i \text{Bel}_j \varphi$ part of the FIPA-S FP (the *relevance precondition*). It simply expresses that asserting an information would be unnecessary if the receiver has already expressed its belief in it. However, our new FP does not demand that group J has no opinion at all about φ , allowing to use *Inform* also to *convince* J in case this group has already expressed its disbelief in φ earlier. $\neg G_K \text{Int}_i G_J \varphi$ in FP effectively demands that agent i did not assert this information using an assertive communication act before, which is also an aspect of the relevance precondition. $\neg G_K \neg G_i \varphi$ ensures that the asserted opinions of agent i are mutually consistent. (The latter is a precondition of *rationality*).

In the IE, $G_K G_i \varphi$ denotes that with asserting φ , it becomes grounded that agent i believes that φ , regardless if she does so privately (i.e., mentally) or not.

As usual we define $\langle i, \text{InformIf}(J, p), K \rangle$ as an abbreviation of $\langle i, \text{Inform}(J, p), K \rangle \cup \langle i, \text{Inform}(J, \neg p), K \rangle$. Hence $\text{Done}(\langle i, \text{InformIf}(J, p), K \rangle) \equiv \text{Done}(\langle i, \text{Inform}(J, p), K \rangle) \vee \text{Done}(\langle i, \text{Inform}(J, \neg p), K \rangle)$.

However, in many cases, we can safely assume that group J immediately starts to publicly believe the asserted information, namely when this group apparently trusts the uttering agent in regard to this information. (A notorious exception are exam situations.) An important particular case is expressed by the following axiom, for $J \subseteq K$ and $\alpha = \langle i, \text{InformIf}(J, \varphi), K \rangle$:⁶

$$(G_K \text{Done}_\alpha \bigwedge_{j \in J} \text{Int}_j \text{Done}(\alpha)) \rightarrow G_K G_J \varphi \quad (1)$$

This specifies that if an agent has requested a certain information before from agent i in form of a closed question (like with “Is it raining

⁶ We here consider that the group J have asked i to publicly declare that φ . (And not each j , as it would be the case if α was $\langle i, \text{InformIf}(\{j\}, \varphi), K \rangle$ in (1).)

outside?"'), it becomes grounded that she believes the answer.⁷

3.2 Request: Requesting an action to be done

Again, we state the FIPA version of the semantics first:

$$\langle i, \text{Request}_{\text{FIPA}}(j, \alpha), K \rangle$$

FP: $FP(\alpha)[i \setminus j] \wedge Bel_i \text{Agent}(j, \alpha) \wedge Bel_i \neg PG_j \text{Done}(\alpha)$

RE: $\text{Done}(\alpha)$

Here, α is an action expression, $FP(\alpha)[i \setminus j]$ denotes the part of the feasibility preconditions of action α where the mental attitudes are those of agent i . $\text{Agent}(j, \alpha)$ states that j is the only agent that ever performs, has performed or will perform α , and $PG_j \text{Done}(\alpha)$ denotes that $\text{Done}(\alpha)$ (i.e., action α has just been performed successfully) is a *persistent goal* [8] of agent j . The RE just specifies that the intended perlocutionary effect of this communication act is to get α done.

Obviously, these specifications again require strong assumptions about mental properties, which are equally problematic as in the case of *Inform*. In addition, $\text{Agent}(j, \alpha)$ reduces the scope of this communication act unnecessarily, disallowing concurrent intention of j to perform the same action herself.

As in our formalism the propositional content of a CA is a formula, a request to do action α is defined as a request that $\text{Done}(\alpha)$ be true. Furthermore, in our case the addressee of a speech act is a group of agents. Thus a request is addressed to each agent of the group in the aim that either at least one agent of the group do the requested action ("open that door"), or each agent of the group do it ("clean that room", addressed to a group of children). $i : \alpha$: denotes that i is the author of action α (making superfluous the FIPA *Agent* predicate). We thus have two kinds of request (whereas there is only one in FIPA):

$$\langle i, \text{RequestSome}(J, J:\alpha), K \rangle \stackrel{\text{def}}{=}$$

$$\langle i, \text{RequestSome}(J, \bigvee_{j \in J} \text{Done}(j:\alpha)), K \rangle$$

FP: $\left(\neg G_K \bigvee_{j \in J} \text{Int}_j \text{Done}(j:\alpha) \right) \wedge \neg G_K \neg \text{Int}_i \left(\bigvee_{j \in J} \text{Done}(j:\alpha) \right)$

IE: $G_K \text{Int}_i \left(\bigvee_{j \in J} \text{Done}(j:\alpha) \right) \wedge G_K \neg G_i \left(\bigvee_{j \in J} \text{Int}_j \text{Done}(j:\alpha) \right)$

So our FP specifies that is not grounded that at least one of the agents in J intends to achieve α already (relevance precondition), and that it is not grounded that agent i does not intend $\text{Done}(\alpha)$ (rationality precondition). The IE is also straightforward: the act results in the grounding that agent i intends that at least one agent in J intends $\text{Done}(\alpha)$ become true, and that i does not believe that one agent in J intends $\text{Done}(\alpha)$.

Second, we define:

$$\langle i, \text{RequestEach}(J, J:\alpha), K \rangle \stackrel{\text{def}}{=}$$

$$\langle i, \text{RequestEach}(J, \bigwedge_{j \in J} \text{Done}(j:\alpha)), K \rangle$$

FP: $\left(\neg G_K \bigwedge_{j \in J} \text{Int}_j \text{Done}(j:\alpha) \right) \wedge \neg G_K \neg \text{Int}_i \left(\bigwedge_{j \in J} \text{Done}(j:\alpha) \right)$

IE: $G_K \text{Int}_i \left(\bigwedge_{j \in J} \text{Done}(j:\alpha) \right) \wedge G_K \neg G_i \left(\bigwedge_{j \in J} \text{Int}_j \text{Done}(j:\alpha) \right)$

which specifies that i intends that each agent of J perform the requested action α . For compatibility reasons, we also define

$$\langle i, \text{Request}(J, \alpha), K \rangle \stackrel{\text{def}}{=}$$

$$\langle i, \text{RequestSome}(J, J:\alpha), K \rangle$$

FIPA also defines the acts *Confirm* (for the confirmation of an uncertain information) and its pendant *Disconfirm* as primitives. But since our *Inform* semantics has an adequately weakened FP that does not require that the asserted information is not uncertain, *Confirm* and *Disconfirm* simply map to *Inform* in our semantics.

4 Case study

In order to demonstrate the properties and the application of our approach, this section presents a brief case study in form of an *agent purchase negotiation* scenario. In particular, we aim to demonstrate the following crucial features of SABS, all not being present in FIPA-S or, by principle, any other BDI-based ACL semantics:

- Pre- and post-conditions of communication acts being only dependent from publicly observable agent behavior, thus being fully verifiable;
- Communication acts with contents being inconsistent with the beliefs and intentions of the participating agents;
- Communication acts addressing groups of agents;
- Multiple communication acts uttered by the same sender, but with mutually inconsistent contents (even towards nested groups);
- Persuasive Inform-acts.

In addition, the example shows how the logging of the grounding state of the negotiation dialogue can replace *commitment stores*, which are usually used to keep track of the various commitments arising during the course of an interaction (like to sell or buy a product). In contrast, by the use of our semantics we obtain the publicly available information about the state of commitment of the participating agents directly in terms of logical post-conditions of communication acts, namely publicly expressed intentions. As explained in Section 1, we consider this to be simpler and formally clear compared to the use of social commitments in the sense of [10].

The interaction roughly follows protocols for *purchase negotiation dialogue games* as known from, e.g., [6], but omitting several details of such protocols which are not relevant for our demonstrative purposes (like the specification of selling options in detail). Also, such protocols often make use of proprietary negotiation locutions, whereas we get along with FIPA-ACL constructs, since in our context, no acts not contained in FIPA-ACL (like the "Promise" and "Threaten" acts in protocols for argumentation-based negotiation) are required. Nevertheless, our scenario is clearly beyond FIPA's *contract net* specification [3].

Our scenario consists of four agents $MAS = \{s_1, s_2, b_1, b_2\}$, representing potential car sellers and customers. In the discourse universe exists two instances θ_1 and θ_2 of some car type θ (e.g., specimen of the Alfa Romeo 159).

We present now the interaction course, consisting of sequential steps in the following form. Note that the interaction course consists of multiple interlaced conversations among different sender/receiver pairs and different overhears (i.e., different "publics" so to say). In particular, agent b_2 is involved in two selling dialogues at the same time.

*Utterance no. sender→receiver: Descriptive act title
Message⁸*

⁷ The intention Int_j can be triggered with FIPA's *QueryIf* act. The schema would work analogously for $\langle i, \text{InformIf}(\{j\}, \neg\varphi), K \rangle$.

⁸ Using syntactical macros according to [4]. Only in case the message primitives are semantically relevant in our context, the respective macros are expanded.

Effect (optionally) gives the effect of the act in terms of grounded formulas, according to SABS and the axioms in Section 2 (so this may go beyond the direct IE).

In contrast, *Private information* (*PI*) optionally unveils relevant mental attitudes before or after an act has been uttered and understood by the respective agents. The PIs are not determined by preceding communication acts, due to agent autonomy. They are also of course usually not available to observers, and just given for explanatory purposes.

U1 $s_1 \rightarrow \{b_1, b_2\}$: Initialize dialogue

$\langle s_1, \text{RequestEach}(\{b_1, b_2\}, \text{enterDialogue}(\theta_1)), \{s_1, b_1, b_2\} \rangle$

U2 $b_1 \rightarrow \{s_1\}$: Enter dialogue

$\langle b_1, \text{Agree}(\{s_1\}, \text{enterDialogue}(\theta_1)), \{s_1, b_1, b_2\} \rangle$

U3 $b_2 \rightarrow \{s_1\}$: Enter dialogue

$\langle b_2, \text{Agree}(\{s_1\}, \text{enterDialogue}(\theta_1)), \{s_1, b_1, b_2\} \rangle$

U4 $s_2 \rightarrow \{b_2\}$: Initialize dialogue

$\langle s_2, \text{Request}(\{b_2\}, \text{enterDialogue}(\theta_2)), \{s_2, b_2\} \rangle$

U5 $b_2 \rightarrow \{s_2\}$: Enter dialogue

$\langle b_2, \text{Agree}(\{s_2\}, \text{enterDialogue}(\theta_2)), \{s_2, b_2\} \rangle$

$P_{I_{s_1}}$: Bel_{s₁} discounts

U6 $s_1 \rightarrow \{b_1, b_2\}$: Information about discount

$\langle s_1, \text{Inform}(\{b_1, b_2\}, \neg \text{discounts}), \{s_1, b_1, b_2\} \rangle$

Effect:

$G_{\{s_1, b_1, b_2\}} G_{s_1} \neg \text{discounts}$

$\wedge G_{\{s_1, b_1, b_2\}} \text{Int}_{s_1} G_{\{b_1, b_2\}} \neg \text{discount}$

Seller s_1 asserts that no discounts can be given while believing ($P_{I_{s_1}}$: Bel_{s₁} discount) that the opposite is true (there might be the company policy that discounts should be given, but that might reduce the seller's individual profit).

U7 $s_1 \rightarrow \{b_2\}$: Information about discount

$\langle s_1, \text{Inform}(\{b_2\}, \text{discounts}), \{s_1, b_2\} \rangle$

Effect:

$G_{\{s_1, b_2\}} G_{s_1} \text{discounts}$

$\wedge G_{\{s_1, b_2\}} \text{Int}_{s_1} G_{b_2} \text{discount}$

While seller s_1 informed group $\{b_1, b_2\}$ that there would be no price discounts, he informs customer b_2 that this is not true (likely because s_1 thinks that b_2 is a valued customer whereas b_1 is not).

U8 $b_2 \rightarrow \{s_1\}$: Query if car type has high accident rate

$\langle b_2, \text{Request}(\{s_1\}, \text{InformIfAccidentRateHigh}), \{s_1, b_2\} \rangle$

Effect:

$G_{\{s_1, b_2\}} \text{Int}_{b_2} \text{Done}(s_1 : \text{InformIfAccidentRateHigh}) \wedge$

$G_{\{s_1, b_2\}} \neg G_{b_2} \text{Int}_{s_1} \text{Done}(s_1 : \text{InformIfAccidentRateHigh}),$ with
 $\text{InformIfAccidentRateHigh} \stackrel{\text{def}}{=}$

$\langle s_1, \text{InformIf}(\{b_2\}, \text{accidentRateHigh}(\theta)), \{s_1, b_2\} \rangle$

$P_{I_{s_1}}$: Bel_{s₁} accidentRateHigh(θ_1)

U9 $s_1 \rightarrow \{b_2\}$: Information about accident rate

$\langle s_1, \text{Inform}(\{b_2\}, \neg \text{accidentRateHigh}(\theta)), \{s_1, b_2\} \rangle$

Effect:

$G_{\{s_1, b_2\}} G_{s_1} \neg \text{accidentRateHigh}(\theta)$

$\wedge G_{\{s_1, b_2\}} G_{b_2} \neg \text{accidentRateHigh}(\theta)$

Note that due to her closed question before and axiom 1 it becomes immediately grounded that b_2 believes the asserted information.

In addition, b_2 privately believes this information also (see PI_{b_2} below), but revises this later.

Seller s_1 asserted $\neg \text{accidentRateHigh}(\theta_1)$ though thinking the opposite.

PI_{b_2} : Bel_{b₂} $\neg \text{accidentRateHigh}(\theta)$

U10 $b_2 \rightarrow \{s_2\}$: Query if car type has high accident rate

$\langle b_2, \text{Request}(\{s_2\}, \text{InformIfAccidentRateHigh}), \{s_2, b_2\} \rangle$

U11 $s_2 \rightarrow \{b_2\}$: Information about accident-damage

$\langle s_2, \text{Inform}(\{b_2\}, \text{accidentRateHigh}(\theta)), \{s_2, b_2\} \rangle$

Again, b_2 publicly believes the information, and trusts it for some reason privately more than the information given by seller s_1 earlier. Nevertheless, $G_{\{s_1, b_2\}} G_{b_2} \neg \text{accidentRateHigh}(\theta_1)$ remains true.

PI_{b_2} : Bel_{b₂} accidentRateHigh(θ)

U12 $b_2 \rightarrow \{s_2\}$: Propose to buy at a certain price

$\langle b_2, \text{Propose}(\{s_2\}, \text{buy}(\theta_2, 10000\mathcal{L})), \{s_2, b_2\} \rangle$

U13 $s_2 \rightarrow \{b_2\}$: Accept proposal

$\langle s_2, \text{AcceptProposal}(\{b_2\}, \text{buy}(\theta_2, 10000\mathcal{L})), \{s_2, b_2\} \rangle$

Effect (with the previous act):

$G_{\{s_2, b_2\}} \text{Int}_{b_2} \text{buy}(\theta_2, 10000\mathcal{L})$ (i.e., b_2 is publicly committed to buy θ_2 at the price of $10000\mathcal{L}$ now).

5 Conclusion

We've proposed a novel approach to the semantics of agent communication, based on verifiable social attitudes which are triggered by observable communication acts. We believe that this approach is more adequate for open systems in comparison both to traditional mentalistic and commitment-based semantics, as it allows to analyze the meaning of messages on the social level without the need to know about mental agent properties or architectural details, while being easily comprehensible, downward compatible to BDI, and fully formalized. A subject of future work in this respect will be the practical application of our approach in the field of interaction protocols, and argumentation and negotiation frameworks.

REFERENCES

- [1] B. F. Chellas, *Modal Logic: an introduction*, Camb. Univ. Press, 1980.
- [2] Philip R. Cohen and Hector J. Levesque, 'Intention is choice with commitment', *Artificial Intelligence*, 42(2–3), 213–261, (1990).
- [3] Foundation for Intelligent Physical Agents. FIPA Interaction Protocol Library Specification, 2000. URL: <http://www.fipa.org/specs/fipa00025>.
- [4] Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, 2002. URL: <http://www.fipa.org/repository/aclspecs.html>.
- [5] B. Gaudou, A. Herzig, and D. Longin, 'Grounding and the expression of belief', in *Proc. 10th Int. Conf. on Princ. of Knowledge Repr. and Reasoning (KR 2006)*, (2006).
- [6] P. McBurney, R. M. van Eijk, S. Parsons, and L. Amgoud, 'A dialogue-game protocol for agent purchase negotiations', *Journal of Autonomous Agents and Multi-Agent Systems*, 7(3), 235–273, (2003).
- [7] M. Nickles, F. Fischer, and G. Weiss, 'Communication attitudes: A formal approach to ostensible intentions, and individual and group opinions', in *Proceedings of the Third International Workshop on Logic and Communication in Multiagent Systems (LCMAS 2005)*, eds., W. van der Hoek and M. Wooldridge, (2005).
- [8] M. D. Sadek, 'A study in the logic of intention', in *Proc. Third Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, eds., Bernhard Nebel, Charles Rich, and William Swartout, pp. 462–473. Morgan Kaufmann Publishers, (1992).
- [9] J. R. Searle, *Speech acts: An essay in the philosophy of language*, Cambridge University Press, New York, 1969.
- [10] M. P. Singh, 'A social semantics for agent communication languages', in *Proceedings of the IJCAI Workshop on Agent Communication Languages*, (2000).
- [11] D. R. Traum, *Computational theory of grounding in natural language conversation*, Ph.D. dissertation, Computer Science Departement, University of Rochester, December 1994.

Contouring of Knowledge for Intelligent Searching for Arguments

Anthony Hunter¹

Abstract. A common assumption for logic-based argumentation is that an argument is a pair $\langle \Phi, \alpha \rangle$ where Φ is a minimal subset of the knowledgebase such that Φ is consistent and Φ entails the claim α . Different logics are based on different definitions for entailment and consistency, and these give us different options for argumentation. For a variety of logics, in particular for classical logic, there is a need to develop intelligent techniques for generating arguments. Since building a constellation of arguments and counterarguments involves repeatedly querying a knowledgebase, we propose a framework based on what we call “contours” for storing information about a knowledgebase that provides boundaries on what is provable in the knowledgebase. Using contours allows for more intelligent searching of a knowledgebase for arguments and counterarguments.

1 INTRODUCTION

Argumentation is a vital aspect of intelligent behaviour by humans. To capture this, there are a number of proposals for logic-based formalisations of argumentation (for reviews see [14, 7]). These proposals allow for the representation of arguments for and against some conclusion, and for attack relationships between arguments. In a number of key examples of argumentation systems, an argument is a pair where the first item in the pair (the support) is a minimal consistent set of formulae that proves the second item (the claim) which is a formula (see for example [3, 10, 4, 1, 11, 5]).

Problemsatically, finding arguments and counterarguments involves much searching of a knowledgebase to ensure they are all found (exhaustiveness), and to ensure each of them has a minimal consistent set of premises entailing the claim (correctness). Proof procedures and algorithms have been developed for finding preferred arguments from a knowledgebase following for example Dung’s preferred semantics (see for example [13, 12, 6, 8, 9]). However, these procedures and algorithms do not offer any ways of ameliorating the cost of repeatedly querying a knowledgebase as part of the process of ensuring exhaustiveness and correctness. In this paper, we address this computational inefficiency by presenting a new technique for intelligently searching a knowledgebase for arguments and counterarguments.

2 LOGICAL ARGUMENTATION

In this section we review an existing proposal for logic-based argumentation [4]. We consider a classical propositional language with deduction denoted by \vdash . We use $\alpha, \beta, \gamma, \dots$ to denote formulae and $\Delta, \Phi, \Psi, \dots$ to denote sets of formulae.

For the following definitions, we first assume a knowledgebase Δ (a finite set of formulae) and use this Δ throughout. We further assume that every subset of Δ is given an enumeration $\langle \alpha_1, \dots, \alpha_n \rangle$ of its elements, which we call its canonical enumeration. This really is not a demanding constraint: In particular, the constraint is satisfied whenever we impose an arbitrary total ordering over Δ . Importantly, the order has no meaning and is not meant to represent any respective importance of formulae in Δ . It is only a convenient way to indicate the order in which we assume the formulae in any subset of Δ are conjoined to make a formula logically equivalent to that subset.

The paradigm for the approach is a large repository of information, represented by Δ , from which arguments can be constructed for and against arbitrary claims. Apart from information being understood as declarative statements, there is no a priori restriction on the contents, and the pieces of information in the repository can be of arbitrary complexity. Therefore, Δ is not expected to be consistent. It need even not be the case that every single formula in Δ is consistent.

The framework adopts a very common intuitive notion of an argument. Essentially, an argument is a set of relevant formulae that can be used to classically prove some claim, together with that claim. Each claim is represented by a formula.

Definition 1. An argument is a pair $\langle \Phi, \alpha \rangle$ such that: (1) $\Phi \subseteq \Delta$; (2) $\Phi \not\vdash \perp$; (3) $\Phi \vdash \alpha$; and (4) there is no $\Phi' \subset \Phi$ such that $\Phi' \vdash \alpha$. We say that $\langle \Phi, \alpha \rangle$ is an argument for α . We call α the **claim** of the argument and Φ the **support** of the argument (we also say that Φ is a support for α).

Example 1. Let $\Delta = \{\alpha, \alpha \rightarrow \beta, \gamma \rightarrow \neg\beta, \gamma, \delta, \delta \rightarrow \beta, \neg\alpha, \neg\gamma\}$. Some arguments are $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$, $\langle \{\neg\alpha\}, \neg\alpha \rangle$, $\langle \{\alpha \rightarrow \beta\}, \neg\alpha \vee \beta \rangle$, and $\langle \{\neg\gamma\}, \delta \rightarrow \neg\gamma \rangle$.

Arguments are not independent. In a sense, some encompass others (possibly up to some form of equivalence). To clarify this requires a few definitions as follows.

Definition 2. An argument $\langle \Phi, \alpha \rangle$ is **more conservative** than an argument $\langle \Psi, \beta \rangle$ iff $\Phi \subseteq \Psi$ and $\beta \vdash \alpha$.

Example 2. $\langle \{\alpha\}, \alpha \vee \beta \rangle$ is more conservative than $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$.

Some arguments directly oppose the support of others, which amounts to the notion of an undercut.

Definition 3. An **undercut** for an argument $\langle \Phi, \alpha \rangle$ is an argument $\langle \Psi, \neg(\phi_1 \wedge \dots \wedge \phi_n) \rangle$ where $\{\phi_1, \dots, \phi_n\} \subseteq \Phi$.

Example 3. Let $\Delta = \{\alpha, \alpha \rightarrow \beta, \gamma, \gamma \rightarrow \neg\alpha\}$. Then, $\langle \{\gamma, \gamma \rightarrow \neg\alpha\}, \neg(\alpha \wedge (\alpha \rightarrow \beta)) \rangle$ is an undercut for $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$. A less conservative undercut for $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$ is $\langle \{\gamma, \gamma \rightarrow \neg\alpha\}, \neg\alpha \rangle$.

¹ UCL Department of Computer Science, London, UK. Email: a.hunter@cs.ucl.ac.uk

Definition 4. $\langle \Psi, \beta \rangle$ is a **maximally conservative undercut** of $\langle \Phi, \alpha \rangle$ iff $\langle \Psi, \beta \rangle$ is an undercut of $\langle \Phi, \alpha \rangle$ such that no undercuts of $\langle \Phi, \alpha \rangle$ are strictly more conservative than $\langle \Psi, \beta \rangle$ (that is, for all undercuts $\langle \Psi', \beta' \rangle$ of $\langle \Phi, \alpha \rangle$, if $\Psi' \subseteq \Psi$ and $\beta \vdash \beta'$ then $\Psi \subseteq \Psi'$ and $\beta' \vdash \beta$).

The value of the following definition of canonical undercut is that we only need to take the canonical undercuts into account. This means we can justifiably ignore the potentially very large number of non-canonical undercuts.

Definition 5. An argument $\langle \Psi, \neg(\phi_1 \wedge \dots \wedge \phi_n) \rangle$ is a **canonical undercut** for $\langle \Phi, \alpha \rangle$ iff it is a maximally conservative undercut for $\langle \Phi, \alpha \rangle$ and $\langle \phi_1, \dots, \phi_n \rangle$ is the canonical enumeration of Φ .

An argument tree describes the various ways an argument can be challenged, as well as how the counter-arguments to the initial argument can themselves be challenged, and so on recursively.

Definition 6. A complete argument tree for α is a tree where the nodes are arguments such that

1. The root is an argument for α .
2. For no node $\langle \Phi, \beta \rangle$ with ancestor nodes $\langle \Phi_1, \beta_1 \rangle, \dots, \langle \Phi_n, \beta_n \rangle$ is Φ a subset of $\Phi_1 \cup \dots \cup \Phi_n$.
3. The children nodes of a node N consist of all canonical undercuts for N that obey 2.

The second condition in Definition 6 ensures that each argument on a branch has to introduce at least one formula in its support that has not already been used by ancestor arguments. This is meant to avoid making explicit undercuts that simply repeat over and over the same reasoning pattern except for switching the role of some formulae (e.g. in mutual exclusion, stating that α together with $\neg\alpha \vee \neg\beta$ entails $\neg\beta$ is exactly the same reasoning as expressing that β together with $\neg\alpha \vee \neg\beta$ entail $\neg\alpha$, because in both cases, what is meant is that α and β exclude each other). As a notational convenience, in examples of argument trees the \diamond symbol is used to denote the claim of an argument when that argument is a canonical undercut (no ambiguity arises as proven in [4]).

Example 4. Let $\Delta = \{\alpha \vee \beta, \alpha \rightarrow \gamma, \neg\gamma, \neg\beta, \delta \leftrightarrow \beta\}$. For this, two argument trees for the claim $\alpha \vee \neg\delta$ are given.

$$\begin{array}{cc} \langle \{\alpha \vee \beta, \neg\beta\}, \alpha \vee \neg\delta \rangle & \langle \{\delta \leftrightarrow \beta, \neg\beta\}, \alpha \vee \neg\delta \rangle \\ \uparrow & \uparrow \\ \langle \{\alpha \rightarrow \gamma, \neg\gamma\}, \diamond \rangle & \langle \{\alpha \vee \beta, \alpha \rightarrow \gamma, \neg\gamma\}, \diamond \rangle \end{array}$$

A complete argument tree is an efficient representation of the counterarguments, counter-counterarguments, ... Furthermore, if Δ is finite, there is a finite number of argument trees with the root being an argument with consequent α that can be formed from Δ , and each of these trees has finite branching and a finite depth (the finite tree property). Note, also the definitions presented in this section can be used directly with first-order classical logic, so Δ and α are from the first-order classical language. Interestingly, the finite tree property also holds for the first-order case [5].

3 MOTIVATION FOR CONTOURING

We now turn to automating the construction of arguments and counterarguments. It is tempting to think that automated theorem proving technology can do more for us than it is guaranteed to. For each argument, we need a minimal set of formulae that proves the claim.

An automated theorem prover (an ATP) may use a “goal-directed” approach, bringing in extra premises when required, but they are not guaranteed to be minimal. For example, supposing we have a knowledgebase $\{\alpha, \alpha \wedge \beta\}$, for proving $\alpha \wedge \beta$, the ATP may start with the premise α , then to prove β , a second premise is required, which would be $\alpha \wedge \beta$, and so the net result is $\{\alpha, \alpha \wedge \beta\} \vdash \alpha \wedge \beta$, which does not involve a minimal set of premises. In addition, an ATP is not guaranteed to use a consistent set of premises since by classical logic it is valid to prove anything from an inconsistency.

So if we seek arguments for a particular claim δ , we need to post queries to an ATP to ensure that a particular set of premises entails δ , that the set of premises is minimal for this, and that it is consistent. So finding arguments for a claim α involves considering subsets Φ of Δ and testing them with the ATP to ascertain whether $\Phi \vdash \alpha$ and $\Phi \not\vdash \perp$ hold. For $\Phi \subseteq \Delta$, and a formula α , let $\Phi?\alpha$ denote a call (a query) to an ATP. If Φ classically entails α , then we get the answer $\Phi \vdash \alpha$, otherwise we get the answer $\Phi \not\vdash \alpha$. In this way, we do not give the whole of Δ to the ATP. Rather we call it with particular subsets of Δ . So for example, if we want to know if $\langle \Phi, \alpha \rangle$ is an argument, then we have a series of calls $\Phi?\alpha, \Phi?\perp, \Phi \setminus \{\phi_1\}?\alpha, \dots, \Phi \setminus \{\phi_k\}?\alpha$, where $\Phi = \{\phi_1, \dots, \phi_k\}$. So the first call is to ensure that $\Phi \vdash \alpha$, the second call is to ensure that $\Phi \not\vdash \perp$, the remaining calls are to ensure that there is no subset Φ' of Φ such that $\Phi' \vdash \alpha$.

We can now summarise all the queries that are posted to the ATP for finding all the arguments for α from the knowledgebase Δ . We summarise it by the set $\text{NaiveQuerying}(\Delta, \alpha)$ defined next.

Definition 7. For a knowledgebase Δ and a formula α ,

$$\text{NaiveQuerying}(\Delta, \alpha) = \{(\Phi?\alpha) \mid \Phi \subseteq \Delta\} \cup \{(\Phi?\perp) \mid \Phi \subseteq \Delta\}$$

Proposition 1. For Δ and α , if $|\Delta| = n$, then $|\text{NaiveQuerying}(\Delta, \alpha)| = 2^{n+1}$.

Clearly, by using all the queries in $\text{NaiveQuerying}(\Delta, \alpha)$, we are taking no account of any of the results that we have gained at any intermediate stage. In other words, we are not being intelligent. Yet if we want to harness automated reasoning, we need principled means for intelligently querying an ATP in order to search a knowledgebase for arguments.

Now when we think of the more difficult problem of not just finding all arguments for α , but then finding all undercuts to these arguments, and by recursion, undercuts to these undercuts. During the course of building an argument tree, there will be repeated attempts to ask the same query, and there will be repeated attempts to ask a query with the same support set. But, importantly, each time a particular subset is tested for a particular claim, we gain more information about Δ . So instead of taking the naive approach of always throwing the result of querying away, we see that this information can be collated about subsets to help guide the search for further arguments and counterarguments. For example, if we know that a particular subset Φ is such that $\Phi \not\vdash \alpha$, and we are looking for an argument with claim $\alpha \wedge \beta$, then we can infer that $\Phi \not\vdash \alpha \wedge \beta$, and there is no need to test Φ for this claim (i.e. it is not useful to make the call $\Phi?\alpha \wedge \beta$).

So as we undertake more tests of Δ for various subsets of Δ and for various claims, we build up a picture of Δ . We can consider these being stored as contours on the (cartographical) map of $\wp(\Delta)$. We formalise this in Section 4.

To make our presentation more concise, from now on, we refer to each subset of Δ by a binary number. Suppose Δ has cardinality n , then we adopt the arbitrary enumeration $\langle \alpha_1, \dots, \alpha_n \rangle$ of Δ , presented in Section 2. Using this, we can then represent any subset Φ of Δ by

a n digit binary number of the form d_1, \dots, d_n . For the i th formula (i.e. α_i) in $\langle \alpha_1, \dots, \alpha_n \rangle$, if α_i is in Φ , then the i th digit (i.e. d_i) in d_1, \dots, d_n is 1, and if α_i is not in Φ , then the i th digit (i.e. d_i) in d_1, \dots, d_n is 0. For example, for the enumeration $\langle \alpha, \beta \wedge \neg\gamma, \gamma \vee \epsilon \rangle$, 000 is $\{\}$, 100 is $\{\alpha\}$, 010 is $\{\beta \wedge \neg\gamma\}$, 001 is $\{\gamma \vee \epsilon\}$, 110 is $\{\alpha, \beta \wedge \neg\gamma\}$, 111 is $\{\alpha, \beta \wedge \neg\gamma, \gamma \vee \epsilon\}$, etc.

Since we will be considering the power set of a knowledgebase, the following definition of the MaxWidth function will be useful for us: If n is $|\Delta|$ and m is $\lfloor n/2 \rfloor$ (i.e. m is the greatest integer less than or equal to $n/2$), then $\text{MaxWidth}(n)$ is $n!/(n-m)!m!$. In other words, of all cardinalities for subsets of Δ , the most numerous are those of cardinality $\lfloor n/2 \rfloor$, and so the MaxWidth function gives the number of subsets of Δ of cardinality $\lfloor n/2 \rfloor$.

4 FRAMEWORK FOR CONTOURING

We start with the ideal situation where we have substantial information about the knowledgebase Δ .

Definition 8. The **contour** for α is $C(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \vdash \alpha \text{ and there is no } \Psi \subset \Phi \text{ such that } \Psi \vdash \alpha\}$, the **uppercontour** for α is $U(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \vdash \alpha\}$, and the **lowercontour** for α , is $L(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \not\vdash \alpha\}$.

Clearly, $C(\alpha) \subseteq U(\alpha)$, and $L(\alpha) \cup U(\alpha)$ is $\wp(\Delta)$, and $L(\alpha) \cap U(\alpha)$ is \emptyset . Furthermore, it is possible that any set in $L(\alpha)$, $C(\alpha)$, or $U(\alpha)$ is inconsistent. Obviously, all the sets in $U(\perp)$ are inconsistent, and none in $L(\perp)$ is inconsistent.

Proposition 2. For any $\Phi \subseteq \Delta$, $\Phi \in C(\alpha)$ iff (1) for all $\Psi \in U(\alpha)$, $\Psi \not\subseteq \Phi$, and (2) for all $\Psi \in L(\alpha)$, $\Phi \not\subseteq \Psi$.

Example 5. For $\langle \alpha \wedge \neg\alpha, \beta, \neg\beta \rangle$, $U(\perp) = \{100, 110, 101, 011, 111\}$, $C(\perp) = \{100, 011\}$, $L(\perp) = \{000, 010, 001\}$, $U(\alpha \vee \beta) = \{100, 010, 101, 110, 011, 111\}$, $C(\alpha \vee \beta) = \{100, 010\}$, and $L(\alpha \vee \beta) = \{000, 001\}$.

We can use contours directly to generate arguments. For this, we obtain $L(\perp)$ from $C(\perp)$ as follows: $L(\perp) = \{\Phi \subseteq \Delta \mid \Phi \subseteq \Psi \text{ and } \Psi \in C(\perp)\}$.

Proposition 3. For any Φ and α , $\Phi \in C(\alpha) \cap L(\perp)$ iff $\langle \Phi, \alpha \rangle$ is an argument.

This means that if we have $C(\alpha)$ and $C(\perp)$, we have all the knowledge we require to generate all arguments for α . By this we mean, we do not need to use the ATP.

However, we may be in a position where we have some γ for which we want to generate arguments, but for which we do not have $C(\gamma)$. In general, we cannot expect to have a contour for every possible claim for which we may wish to construct an argument. To address this, we will require an ATP, but we can focus our search for the arguments, so that we can reduce the number of calls that we make to the ATP. For this we can identify Boolean constituents of γ for which we do have the contours. To support this, we require the following definition.

Definition 9. For any α, β , the **contour conjunction** and **contour disjunction** operators, denoted \oplus and \otimes respectively, are defined as follows.

$$\begin{aligned} C(\alpha) \oplus C(\beta) &= C(\alpha) \cup C(\beta) \cup \{\Phi \cup \Psi \mid \Phi \in C(\alpha) \& \Psi \in C(\beta)\} \\ C(\alpha) \otimes C(\beta) &= C(\alpha) \cup C(\beta) \cup \{\Phi \cap \Psi \mid \Phi \in C(\alpha) \& \Psi \in C(\beta)\} \end{aligned}$$

Example 6. Consider $\langle \alpha, \beta, \alpha \vee \beta \rightarrow \gamma, \alpha \vee \beta \rightarrow \delta \rangle$. So $C(\gamma) = \{1010, 0110\}$ and $C(\delta) = \{1001, 0101\}$. Hence,

$$\begin{aligned} C(\gamma) \oplus C(\delta) &= \{1011, 0111, 1111, 1010, 0110, 1001, 0101\} \\ C(\gamma) \otimes C(\delta) &= \{1000, 0000, 0100, 1010, 0110, 1001, 0101\} \end{aligned}$$

Note, $C(\gamma \wedge \delta) = \{1011, 0111\}$ and $C(\gamma \vee \delta) = \{1010, 0110, 1001, 0101\}$.

Clearly, the \oplus and \otimes operators are associative and commutative. We also obtain the following containment results.

Proposition 4. For any α, β , $C(\alpha \wedge \beta) \subseteq (C(\alpha) \oplus C(\beta))$

Proposition 5. For any α, β , $C(\alpha \vee \beta) \subseteq (C(\alpha) \otimes C(\beta))$

So if we are looking for an argument for $\alpha \wedge \beta$, we look in the set $C(\alpha) \oplus C(\beta)$. Similarly if we are looking for an argument for $\alpha \vee \beta$, we look in the set $C(\alpha) \otimes C(\beta)$. Furthermore, the number of sets to consider in $C(\alpha) \oplus C(\beta)$, and similarly in $C(\alpha) \otimes C(\beta)$, is a quadratic function of the number of sets in each of $C(\alpha)$ and $C(\beta)$.

Proposition 6. For any α and β , if $|C(\alpha)| = p$ and $|C(\beta)| = q$, then $|C(\alpha) \oplus C(\beta)| \leq pq + p + q$ and $|C(\alpha) \otimes C(\beta)| \leq pq + p + q$.

Now we consider an operator to facilitate the use of a contour $C(\alpha)$ to find arguments with the claim $\neg\alpha$.

Definition 10. For any $X \subseteq \wp(\Delta)$, the **shift** operator, denoted \div , is defined as follows.

$$\div X = \{\Phi \subseteq \Delta \mid \text{for all } \Psi \in X, \Phi \not\subseteq \Psi \text{ and } \Psi \not\subseteq \Phi\}$$

Example 7. Consider $\langle \alpha \vee \beta, \neg\alpha, \neg\beta, \neg\gamma \wedge \delta, \neg\delta \wedge \beta \rangle$. So $C(\beta)$ is $\{11000, 00001\}$. Hence, $\div C(\beta)$ is $\{10100, 10010, 10110, 01100, 01010, 01110, 00100, 00010, 00110\}$. By comparison, $C(\neg\beta)$ is $\{00100\}$.

Whilst \div is, in a weak sense, a kind of complementation operator, properties such as $\div(\div C(\alpha)) = C(\alpha)$ do not hold. But we do have the following useful property which shows how given $C(\alpha)$, we can use $\div C(\alpha)$ to focus our search for an argument for $\neg\alpha$.

Proposition 7. For any α , $(C(\neg\alpha) \cap L(\perp)) \subseteq (\div C(\alpha) \cap L(\perp))$

We conclude this section by considering undercuts. For any undercut, the claim is the negation of the support of its parent, and the support of the parent is some subset of Δ . Suppose the support of the parent is $\{\delta_1, \dots, \delta_k\}$, so the claim of any undercut is $\neg(\delta_1 \wedge \dots \wedge \delta_k)$. This claim is equivalent to $\neg\delta_1 \vee \dots \vee \neg\delta_k$. So, if we have contours for each of $\neg\delta_1, \dots, \neg\delta_k$, we can focus our search for any these undercut in the space delineated by $C(\neg\delta_1) \otimes \dots \otimes C(\neg\delta_k)$. So we may consider keeping a contour for the negation of each element in Δ .

5 USING CONTOURS

We now consider how contours can improve our use of an ATP by using fewer queries than NaiveQuerying(Δ, α). For a knowledgebase Δ , a formula α , and a set of contours Θ , if $C(\alpha) \in \Theta$, then we know it is not necessary to make any calls to the ATP.

If we are looking for arguments with a claim that is a conjunction of formulae for which we have contours, the queries to the ATP are delineated by the following function.

$$\begin{aligned} \text{ConjunctionQuerying}(\{C(\alpha), C(\beta), C(\perp)\}, \alpha \wedge \beta) \\ = \{(\Phi? \alpha) \mid \Phi \in C(\alpha) \oplus C(\beta) \text{ and } \Phi \in L(\perp)\} \end{aligned}$$

Similarly, if we are looking for arguments with a claim that is a disjunction of formulae for which we have contours, the queries to the ATP are delineated by the following function.

$$\begin{aligned} \text{DisjunctionQuerying}(\{\mathcal{C}(\alpha), \mathcal{C}(\beta), \mathcal{C}(\perp)\}, \alpha \vee \beta) \\ = \{(\Phi? \alpha) \mid \Phi \in \mathcal{C}(\alpha) \otimes \mathcal{C}(\beta) \text{ and } \Phi \in \mathcal{L}(\perp)\} \end{aligned}$$

Proposition 8. For Δ , $\alpha \wedge \beta$, and $\alpha \vee \beta$, if $|\mathcal{C}(\alpha)| = p$ and $|\mathcal{C}(\beta)| = q$ and $r = pq + p + q$ then

$$\begin{aligned} |\text{ConjunctionQuerying}(\{\mathcal{C}(\alpha), \mathcal{C}(\beta), \mathcal{C}(\perp)\}, \alpha \wedge \beta)| \leq r \\ |\text{DisjunctionQuerying}(\{\mathcal{C}(\alpha), \mathcal{C}(\beta), \mathcal{C}(\perp)\}, \alpha \vee \beta)| \leq r \end{aligned}$$

Now we turn to finding undercuts which can be a substantial part of the cost of constructing an argument tree, and indeed can involve many times more work than just finding the root of an argument tree. If we have a contour for the negation of each formula in Δ , then we can use the following delineation on queries.

$$\begin{aligned} \text{UndercutQuerying}(\{\mathcal{C}(\neg\delta_1), \dots, \mathcal{C}(\neg\delta_k), \mathcal{C}(\perp)\}, \neg\delta_1 \vee \dots \vee \neg\delta_k) \\ = \{(\Phi? \alpha) \mid \Phi \in \mathcal{C}(\neg\delta_1) \otimes \dots \otimes \mathcal{C}(\neg\delta_k) \text{ and } \Phi \in \mathcal{L}(\perp)\} \end{aligned}$$

Example 8. For $\langle \alpha, \beta, \delta \rightarrow (\neg\alpha \vee \neg\beta), \delta, \gamma, \gamma \rightarrow \neg\delta \rangle$, $\mathcal{C}(\neg\alpha) = \{011100, 000111\}$, $\mathcal{C}(\neg\beta) = \{101100, 000111\}$, $\mathcal{C}(\neg(\delta \rightarrow (\neg\alpha \vee \neg\beta))) = \{110100, 000111\}$, $\mathcal{C}(\neg\delta) = \{000011\}$, $\mathcal{C}(\neg\gamma) = \{111100, 000101\}$, $\mathcal{C}(\neg(\gamma \rightarrow \neg\delta)) = \{111100, 000110\}$, and $\mathcal{C}(\perp) = \{111100, 000111\}$. Consider undercuts for $\langle 110000, \alpha \wedge \beta \rangle$, i.e. those with the claim $\neg\alpha \vee \neg\beta$. So $\text{UndercutQuerying}(\{\mathcal{C}(\neg\alpha), \mathcal{C}(\neg\beta), \mathcal{C}(\perp)\}, \neg\alpha \vee \neg\beta) = \{001100? \neg\alpha \vee \neg\beta, 011100? \neg\alpha \vee \neg\beta, 101100? \neg\alpha \vee \neg\beta\}$. By comparison, $\mathcal{C}(\neg\alpha \vee \neg\beta) = \{001100\}$.

Proposition 9. For Δ and α , if $|\Delta| = n$, then $0 \leq |\text{UndercutQuerying}(\{\mathcal{C}(\neg\delta_1), \dots, \mathcal{C}(\neg\delta_k), \mathcal{C}(\perp)\}, \neg\delta_1 \vee \dots \vee \neg\delta_k)| \leq 2^{n-2} + 1$.

The best case for the UndercutQuerying function is when $k = 1$. This is when the argument being undercut has a support with just one premise. So the undercut has to just undercut this premise, and therefore, the undercut can be obtained directly from the relevant contour $\mathcal{C}(\neg\delta_1)$ without recourse to the ATP. The worst case for the UndercutQuerying function is when $|\Delta| = n$ and $k = (\text{MaxWidth}(n) - 1)$, and there is a $\Phi_1 \in \mathcal{C}(\neg\delta_1)$ and .. and a $\Phi_k \in \mathcal{C}(\neg\delta_k)$ such that for each $\Phi_i \in \{\Phi_1, \dots, \Phi_k\}$, $|\Phi_i| = \lfloor n/2 \rfloor$, and for all $\Phi_i, \Phi_j \in \{\Phi_1, \dots, \Phi_k\}$, $\Phi_i \neq \Phi_j$. The worst case is clearly an extreme situation and it would seem that on average the number of queries raised by the UndercutQuerying function would be significantly lower, as can be seen for lower values of k using iterated applications of Proposition 8.

6 PARTIAL CONTOURS

Let Π be a set of answers to queries to the ATP. In other words, for a series of queries $(\Phi_1? \alpha_1), \dots, (\Phi_n? \alpha_n)$, we have obtained a set of answers $\Pi = \{\pi_1, \dots, \pi_n\}$, where for each $\pi_i \in \Pi$, the answer is either of the form $\Phi_i \vdash \alpha_i$ or of the form $\Phi_i \not\vdash \alpha_i$. Using Π , we want to generate a partial contour $\mathcal{C}(\Pi, \alpha)$ for α . We will define it so that $\mathcal{C}(\Pi, \alpha)$ is a subset of $\mathcal{C}(\alpha)$ calculated on the basis of Π .

Definition 11. For Π and α , the **partial uppercontour**, is $\mathcal{U}(\Pi, \alpha) = \{\Phi \subseteq \Delta \mid \Psi \subseteq \Phi \text{ and } (\Psi \vdash \alpha) \in \Pi\}$ and the **partial lowercontour**, is $\mathcal{L}(\Pi, \alpha) = \{\Phi \subseteq \Delta \mid \Phi \subseteq \Psi \text{ and } (\Psi \not\vdash \alpha) \in \Pi\}$.

For $\Pi \subseteq \Pi'$, $\mathcal{U}(\Pi, \alpha) \subseteq \mathcal{U}(\Pi', \alpha)$ and $\mathcal{L}(\Pi, \alpha) \subseteq \mathcal{L}(\Pi', \alpha)$.

Definition 12. For Π , the **partial contour** for $\mathcal{C}(\Pi, \alpha)$ is $\mathcal{C}(\Pi, \alpha) = \{\Phi \in \mathcal{U}(\Pi, \alpha) \mid \text{for all } \phi \in \Phi, (\Phi \setminus \{\phi\}) \in \mathcal{L}(\Pi, \alpha)\}$.

So from Π , we construct the partial uppercontour and the partial lowercontour for some α , and then from these we can construct the partial contour for α .

Example 9. For $\langle \alpha, \neg\alpha, \beta \rangle$, let $\Pi = \{(111 \vdash \perp), (101 \not\vdash \perp), (011 \not\vdash \perp), (100 \not\vdash \perp), (110 \vdash \perp), (010 \not\vdash \perp)\}$. Hence, $\mathcal{C}(\Pi, \perp) = \{110\}$.

We use partial contours in the same way as we use full contours. The only proviso is that with partial contours, we have incomplete information about the knowledgebase. So for example, since $\mathcal{C}(\Pi, \alpha) \subseteq \mathcal{C}(\alpha)$, we are not guaranteed to obtain all arguments for α from just using $\mathcal{C}(\Pi, \alpha)$. However, for any $\Phi \in \mathcal{C}(\Pi, \alpha)$, we know that Φ or any superset of Φ implies α , and any proper subset does not imply Φ . So any element in $\mathcal{C}(\Pi, \alpha)$ can have a profound effect on searching for arguments. We use the shift operator to formalise this.

Proposition 10. If $\langle \Phi, \alpha \rangle$ is an argument, then $\Phi \in ((\mathcal{C}(\Pi, \alpha) \cup \div\mathcal{C}(\Pi, \alpha))$.

Proposition 11. If $\Phi \in \mathcal{C}(\Pi, \alpha) \cap \mathcal{L}(\perp)$, then $\langle \Phi, \alpha \rangle$ is an argument.

We can delineate the queries sent to the ATP when searching for an argument for α , with a partial contour $\mathcal{C}(\Pi, \alpha)$, using the following definition, where $\mathcal{M}(\Pi, \perp)$ is $\mathcal{L}(\Pi, \perp) \cup \mathcal{U}(\Pi, \perp)$.

$$\begin{aligned} \text{PartialQuerying}(\{\mathcal{C}(\Pi, \alpha), \mathcal{C}(\Pi, \perp)\}, \alpha) = \\ \{(\Phi? \alpha) \mid \Phi \in \div\mathcal{C}(\Pi, \alpha) \text{ and } \Phi \notin \mathcal{U}(\Pi, \perp)\} \\ \cup \{(\Phi? \perp) \mid \Phi \in (\div\mathcal{C}(\Pi, \alpha) \cup \mathcal{C}(\Pi, \alpha)) \text{ and } \Phi \notin \mathcal{M}(\Pi, \perp)\} \end{aligned}$$

The utility of a partial contour is dependent on the membership of Π . However, we do have $\text{PartialQuerying}(\{\mathcal{C}(\Pi, \alpha), \mathcal{C}(\Pi, \perp)\}, \alpha)$ being a subset of $\text{SimpleQuerying}(\Delta, \mathcal{C}(\perp), \alpha)$ and as Π increases, the difference is increasingly marked. So in any case, whatever is in a partial contour can reduce the number of calls to the ATP.

Now we turn to the question of what are the bounds on the number of queries (i.e. size of Π) to build a contour. In the worst case, for any α , to ensure $\mathcal{C}(\Pi, \alpha) = \mathcal{C}(\alpha)$, it is necessary for Π to have 2^n queries where $\Pi = \{(\Phi? \alpha) \mid \Phi \subseteq \Delta\}$. However, we can take a more intelligent approach to decrease Π . For example, if we generate Π dynamically, once we have enough information from Π to add a particular $\Phi \subseteq \Delta$ to a particular contour, we have no need to seek supersets or subsets of Φ for that contour. Furthermore, we can envisage that the contours are built over time, as a by-product of using a knowledgebase. So each time the ATP is queried, the answer to the query is added to Π . As items are added to Π , the contours are constructed incrementally. We can therefore think of contours as being formed by a kind of lemma generation.

Finally, if we use partial contours, we do not even need to assume that the knowledgebase is fixed, since every partial contour of a knowledgebase Δ is a partial contour of a knowledgebase $\Delta \cup \Delta'$ for any Δ' .

7 STORING CONTOURS

So far we have made a case for the value of using contours. Even in the worst case, they offer an improvement over naive searching for arguments and counterarguments. By maintaining appropriate contours, the numbers of calls to the ATP is always decreased. The downside to maintaining contours, or even partial contours, is the amount of information that needs to be kept about the knowledgebase over time.

Proposition 12. For any α , if $|\Delta| = n$, then $1 \leq C(\alpha) \leq \text{MaxWidth}(n)$.

So for example, if the cardinality of our knowledgebase is 10, we may in the worst case, have a cardinality for $C(\perp)$ of 252. This is not surprising given that $C(\perp)$ is the set of minimally inconsistent subsets of Δ . Of course, this is a worst case scenario, and it would indicate a remarkably inconsistent knowledgebase for which it would be difficult to imagine arising in the real-world. Nonetheless, it raises the question of whether we could compromise on the information we retain from Π , and yet still reduce the number of queries to the ATP. We address this next.

The following result suggests that a simple solution is that when a contour or partial contour is being constructed, it appears to be too large to be manageable, it may be better to erase it, and construct the elements of the contour as and when required, and furthermore, if it is sufficiently large, it is relatively straightforward to find them because they are the subsets of Δ of cardinality $\lfloor n/2 \rfloor$.

Proposition 13. Let $|\Delta| = n$. For any α , as the cardinality of $C(\alpha)$ increases towards $\text{MaxWidth}(n)$, the average size of the elements of $C(\alpha)$ approaches $\lfloor n/2 \rfloor$.

As another alternative to storing all the precise information we have about the contours, given some Π , we propose using outline contours. To do this, we require the subsidiary notion of a **k -partition**: For a set $\Lambda = \{\Phi_1, \dots, \Phi_n\}$, where $k \leq n$, a k -partition is a partitioning of Λ into k disjoint subsets of Λ such that the partition with most members has at most one member more than the partition with fewest members. So a k -partition is as close as possible to a partitioning with partitions of equal size. Because a k -partition is a partitioning, each element of Λ is in exactly one partition. We also require the subsidiary notion of a **serial union**: For a set $\Lambda = \{\Phi_1, \dots, \Phi_n\}$, where $k \leq n$, suppose $(\Lambda_1, \dots, \Lambda_k)$ is a k -partitioning of Λ . Then the serial union of $(\Lambda_1, \dots, \Lambda_k)$ is $(\cup(\Lambda_1), \dots, \cup(\Lambda_k))$.

Definition 13. For a set of answers Π , a formula α , and a threshold $k \in \mathbb{N}$, an **outline contour** $C(\Pi, \alpha, k)$ is defined as follows: If $|C(\Pi, \alpha)| < k$, then $C(\Pi, \alpha, k) = C(\Pi, \alpha)$, otherwise let $C(\Pi, \alpha, k)$ be the serial union of a k -partitioning of $C(\Pi, \alpha)$.

So if $C(\Pi, \alpha, k)$ is $C(\Pi, \alpha)$, then we are keeping the explicit information we have about the contour. But, if $|C(\Pi, \alpha)| > k$, then we merge sets in $C(\Pi, \alpha)$ so that we never have more than k sets. In any case, we can recover the contours from each outline contour.

Proposition 14. If $C(\Pi, \alpha) = C(\alpha)$, and $k = 1$, then $C(\Pi, \alpha, k) = \{\cup C(\alpha)\}$.

Example 10. For $\langle \alpha, \neg\alpha, \beta \wedge \neg\beta, \neg\alpha \wedge \gamma, \alpha \vee \beta, \neg\beta \rangle$,

$$C(\perp) = \{110000, 001000, 100100, 010011\}$$

Let Π be such that $C(\Pi, \perp) = C(\perp)$ and let $k = 2$. So there are four possibilities for $C(\Pi, \perp, k)$. One of them is $\{111000, 110111\}$.

Proposition 15. Let Π be such that $C(\Pi, \alpha) = C(\alpha)$. For any k , if $\langle \Phi, \alpha \rangle$ is an argument, then there is a $\Psi \in C(\Pi, \alpha, k)$, such that $\Phi \subseteq \Psi$.

So outline contours involve more queries to the ATP than contours, but less space is required to store them.

8 DISCUSSION

Much progress has been made in developing formalisms for argumentation. Some algorithms for argumentation have been developed (for example [12, 6, 2, 11]). However, relatively little progress has been made in developing techniques for overcoming the computational challenges of constructing arguments, and in particular for intelligent searching for arguments.

In this paper, we have (1) clarified the need to manage the querying of an ATP when constructing arguments and counterarguments; (2) introduced contours as a way of representing information about a knowledgebase obtained by querying with an ATP; (3) shown how we can construct arguments and counterarguments using contours and shown how this is more efficient than naively using the knowledgebase with the ATP to search for arguments; (4) shown how we can construct contours incrementally; and (5) considered a couple of simple strategies for offsetting the cost of maintaining larger contours. From the theoretical framework for contouring presented in this paper, it should be straightforward to develop algorithms for harnessing an ATP and undertaking empirical evaluation.

Whilst our presentation is based on a particular approach to logic-based argumentation, we believe the proposal could easily be adapted for a range of other logic-based approaches to argumentation (for example [11, 1, 9]).

REFERENCES

- [1] L Amgoud and C Cayrol, ‘A model of reasoning based on the production of acceptable arguments’, *Annals of Mathematics and Artificial Intelligence*, **34**, 197–216, (2002).
- [2] P Baroni and M Giacomin, ‘Argumentation through a distributed self-stabilizing approach’, *Journal of Experimental and Theoretical Artificial Intelligence*, **14**(4), 273–301, (2002).
- [3] S Benferhat, D Dubois, and H Prade, ‘Argumentative inference in uncertain and inconsistent knowledge bases’, in *Proceedings of Uncertainty in Artificial Intelligence*, pp. 1449–1445. Morgan Kaufmann, (1993).
- [4] Ph Besnard and A Hunter, ‘A logic-based theory of deductive arguments’, *Artificial Intelligence*, **128**, 203–235, (2001).
- [5] Ph Besnard and A Hunter, ‘Practical first-order argumentation’, in *Proc. of the 20th National Conference on Artificial Intelligence (AAAI’2005)*, pp. 590–595. MIT Press, (2005).
- [6] C Cayrol, S Doutre, and J Mengin, ‘Dialectical proof theories for the credulous preferred semantics of argumentation frameworks’, in *Quantitative and Qualitative Approaches to Reasoning with Uncertainty*, volume 2143 of *LNCS*, pp. 668–679. Springer, (2001).
- [7] C Chesñevar, A Maguitman, and R Loui, ‘Logical models of argument’, *ACM Computing Surveys*, **32**, 337–383, (2001).
- [8] Y Dimopoulos, B Nebel, and F Toni, ‘On the computational complexity of assumption-based argumentation for default reasoning’, *Artificial Intelligence*, **141**, 57–78, (2002).
- [9] P Dung, R Kowalski, and F Toni, ‘Dialectic proof procedures for assumption-based, admissible argumentation’, *Artificial Intelligence*, (2005). (in press).
- [10] M Elvang-Gøransson, P Krause, and J Fox, ‘Dialectic reasoning with classically inconsistent information’, in *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 114–121. Morgan Kaufmann, (1993).
- [11] A García and G Simari, ‘Defeasible logic programming: An argumentative approach’, *Theory and Practice of Logic Programming*, **4**(1), 95–138, (2004).
- [12] A Kakas and F Toni, ‘Computing argumentation in logic programming’, *Journal of Logic and Computation*, **9**, 515–562, (1999).
- [13] H Prakken and G Sartor, ‘Argument-based extended logic programming with defeasible priorities’, *Journal of Applied Non-Classical Logics*, **7**, 25–75, (1997).
- [14] H Prakken and G Vreeswijk, ‘Logical systems for defeasible argumentation’, in *Handbook of Philosophical Logic*, ed., D Gabbay, Kluwer, (2000).

On the Inability of Gathering by Asynchronous Mobile Robots with Initial Movements¹

Branislav Katreňiak² and Jana Katreňiaková²

Abstract. Consider a community of simple autonomous robots freely moving in the plane. The robots are decentralized, asynchronous, deterministic without the common coordination system, identities, direct communication, memory of the past, but with the ability to sense the positions of the other robots. Existing results show the inability of gathering in this model and solve the gathering problem in the model extended by the multiplicity detection capability. However, the previous results rely on the fact that the robots are initially not moving. We prove that the gathering problem is unsolvable in the model with initial movement vectors even with the multiplicity detection capability.

1 INTRODUCTION

We consider a distributed system populated by a set of autonomous mobile robots. The robots are anonymous, have no common knowledge, no common sense of the direction (e.g. compass), no central coordination and no means of direct communication. The behavior of these robots is quite simple, each of them idles, senses, computes and moves in a cycle. In particular, each robot is capable of sensing the positions of the other robots with respect to its local coordination system, performing local computations on the observed data and moving towards the computed destination. The local computation is done according to a deterministic algorithm that takes only the sensed robots' positions as input and returns a destination point towards which the executing robot moves. All robots perform the same algorithm. The main research in this area is focused on understanding under which conditions are these robots able to complete given tasks, e.g. exploring the plane, forming particular patterns, gathering in a single point, etc.

The convergence problem [4] (robots have to converge together) and the problem of forming the biangular circle [7] can be solved in this basic model, however the gathering problem cannot [8]. Various extensions of the model have been proposed to increase its power. Adding the common sense of the orientation (compass) made the problem of arbitrary pattern formation solvable [6] and the gathering problem (robots have to meet at one point in finite time) became solvable even when robots were restricted to the limited visibility [5]. Another studied extension was the multiplicity detection and the gathering problem became solvable with this capability [2].

However, all the previous results supposed that the robots are initially not moving. Especially the algorithms in [2], [7] used the assumption very extensively to prevent certain configurations (robots'

positions together with their movement vectors) in order to infer properties about the robots' movement from the observed robots' positions.

In this paper we relax this assumption and allow the robots to have arbitrary initial movement vectors. This assumption allows for better sequential composition of algorithms and makes the model even more selfstabilizing – it does not only simulate errors causing robots to stop but also a finite number of errors causing robots to miss the destination point computed by the algorithm e.g. by external push (Theorem 2).

After the model definition in Section 2 we mention some basic properties of the introduced model. In Section 4 we analyze the solvability of the convergence problem and in Section 5 we present the proof that the gathering problem is insolvable for three robots even with the multiplicity detection capability and conclude that the basic model with the multiplicity detection and the model with initial vectors and the multiplicity detection are different.

2 MODEL

We use the model introduced in [6] weakened by canceling the implicit zero initial vectors assumption. The robots do not start synchronously from the waiting state but in the moving phase with some arbitrary initial destination point. Such a model will be called model with initial vectors.

2.1 Model definition

Each robot is viewed as a point in a plane equipped with sensors. It can observe the set of all points which are occupied by at least one other robot. Note that the robot only knows whether there are any other robots at a specific point, but it has no knowledge about their number (i.e. it cannot tell how many robots are at a given location). The *local view* of each robot consists of a unit of length, an origin (w.l.o.g. the position of the robot in its current observation), an orientation of angles and the coordinates of the observed points. No kind of agreement on the unit of length, the origin or the orientation of angles is assumed among the robots.

A robot is initially *moving* towards some initial destination point. When the robot reaches this destination, it switches to the basic *waiting* state (*Wait*). Asynchronously and independently from the other robots, it *observes* the environment (*Look*) by activating its sensors. The sensors return a snapshot of the world, i.e. the set of all points occupied by at least one other robot, with respect to the local coordinate system. Then, based only on its local view of the world, the robot *calculates* its destination point (*Compute*) according to its deterministic algorithm (the same for all robots). After the computation

¹ Supported by VEGA 1/3106/06

² Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava
e-mail {katreňiak, katreňiaková}@dcs.fmph.uniba.sk

the robot *moves* towards its destination point (*Move*); if the destination point is the current location, the robot stays on its place. A move may stop before the robot reaches its destination (e.g. because of limits to the robot's motion energy). The robot then returns to the waiting state. The sequence *Wait – Look – Compute – Move* forms a *cycle* of a robot.

The robots are *fully asynchronous*, the amount of time spent in each phase of a cycle (even in the movement towards the initial destination point) is finite but otherwise unpredictable. In particular, the robots do not have a common notion of time. As a result, robots can be seen by the other robots while moving, and thus computations can be made based on obsolete observations. The robots are *oblivious*, meaning that they do not remember any previous observations nor computations performed in any previous cycles. The robots are *anonymous*, meaning that they are indistinguishable by their appearance, and they do not have any kind of identifiers that can be used during the computation. Finally, the robots have *no means of direct communication*: any communication occurs in a totally implicit manner, by observing the other robots' positions.

There are two limiting assumptions concerning *infinity*:

Assumption 1: The amount of time required by a robot to complete a cycle is finite and bounded from above by a fixed constant.

Assumption 2: The distance traveled by a robot in a cycle is bounded from below by a fixed ε (or the robot gets to the destination point).

As no other assumptions on space exist, the distance traveled by a robot in a cycle is unpredictable.

One considered capability of robots' sensors is the multiplicity detection capability. Sensors of such robots do not return only the set of occupied positions but also the information about the multiplicity at each occupied position. The multiplicity can be expressed either as a number of robots at particular position or only as a binary information one robot/more robots. We will consider the stronger variant with full information.

3 MODEL PROPERTIES

We defined the initial movement of robots as uninterruptable. But the model does not change if we allow them to be interruptable.

Theorem 1 *The model with initial vectors is equal to its modification where the initial movements can be interrupted as the movements in the live cycles.*

From the point of view of any robot's algorithm, the initial destination point is unpredictable. If the initial movement has to be interrupted at some point P , the initial destination point could be defined as this point P .

We define a *push* as an asynchronous event which sets some robot into the moving state towards an arbitrary destination point.

We define the model with finite number of pushes as the model in which finite number of pushes is allowed.

Theorem 2 *The model with the initial vectors is equal to its extension with finite number of pushes and is also equal to the basic model (with zero initial vectors) with finite number of pushes.*

In the models with finite number of pushes we execute the computation until the last push occurs. At this point we construct exactly the same configuration (robots' algorithms, positions and movement vectors) as in the model with initial vectors. Computation will continue exactly the same way because no more pushes can occur.

Models with finite number of pushes can simulate the model with initial vectors by pushing the robots at the start to give them the same initial vectors.

4 CONVERGENCE

Convergence problem orders robots to converge together but they don't have to meet at one point in finite time.

[Convergence] Given is a group of n robots in the plane. Find an algorithm for the robots such that for every $\varepsilon > 0$ there exist time t that after this time the distance of every two robots will be at most ε .

The convergence problem can be solved by easy center of gravity algorithm. Denote by $r_i[t]$ the location of robot i at time t . The algorithm *Center.Of.Gravity* is very simple. After observing the current configuration at some time t , the robot i computes as its goal point ($c_i[t]$) the average location of all robot positions (including its own) and moves towards the calculated goal point. A formal definition of this algorithm follows.

Algorithm Center.Of.Gravity (code for robot i at time t)

1. Calculate the center of gravity $c_i[t] = \frac{1}{N} \sum_j r_j[t]$
2. Move to the point $c_i[t]$

It was proved in [4] that the algorithm *Center.Of.Gravity* solves the convergence problem in the basic model and also in the model with initial vectors.

5 GATHERING

Gathering problem orders robots to meet at any point in the plane in finite time.

[Gathering] Given is a group of n robots in the plane. Find an algorithm for the robots in order to gather all robots at one point in finite time. The gathering point is not determined before.

It was shown in [8] that gathering is unsolvable in the basic model without any extensions. If we give robots the multiplicity detection capability, gathering becomes solvable for at least three robots [2],[3]. We show that in the model with initial vectors the multiplicity detection capability does not help and gathering stays unsolvable for three robots even with multiplicity detection extension.

To show the insolvability of the gathering problem, we will choose for every possible (fixed) algorithm such initial robots' positions, initial robots' vectors, robots' local coordination systems and the scheduler that prevent the robots from gathering (in finite time). As many events and actions in the model are asynchronous from the robots' algorithms' point of view, we can schedule them arbitrary (only restricted by Assumptions 1 and 2) in our scheduler.

We define a *similar projection* as the projection that preserves ratios of distances and angles. Two configurations are similar when one can be obtained from the other by a similar projection. Similar configurations are indistinguishable from the robots' algorithms' point of view (assuming the robots' positions and movement vectors being projected too).

Usual way of preventing the robots from gathering is to force them to repeat any similar configuration during the model computation. Then the sequence between the two similar configurations can be repeated indefinitely many times and robots will never gather at one point in finite time. Scheduler can arbitrary slower and fasten the robots' movement but it cannot interrupt the movement (Assumption 2).

The symmetrical configuration forces the algorithms to act equally (all the same) in some situations.

Theorem 3 For every algorithm solving the gathering problem holds that if the equilateral triangle configuration is observed, algorithm has to choose the center of the triangle as the destination point.

Suppose that for the equilateral triangle the algorithm does not choose the center of the triangle as the destination point. We choose the initial robots' positions as the equilateral triangle, zero initial vectors, and synchronous scheduler. Local coordination system are chosen such that all robots observe the same situation (we use the symmetry of the situation).

As the robots are indistinguishable, every robot must choose the same (symmetrical) destination vector. After one step of the synchronous scheduler we get a similar configuration.

Note that Theorem 3 holds even when robots are equipped with the multiplicity detection capability.

Having Theorem 3 in mind we are going to find the mentioned initial positions, initial vectors and scheduler that will force the robots to reach the similar positions again.

We represent the robots positions (points in the plane) by complex numbers. We choose the initial positions of robots to form an equilateral triangle $A = 1 + 0i$, $B = -\frac{1}{2} - \frac{\sqrt{3}}{2}i$, $C = -\frac{1}{2} + \frac{\sqrt{3}}{2}i$ with center $S = 0 + 0i$ and mark the initial movement vector of robot A as \vec{a} , robot B as \vec{b} and robot C as \vec{c} (see Fig.1).

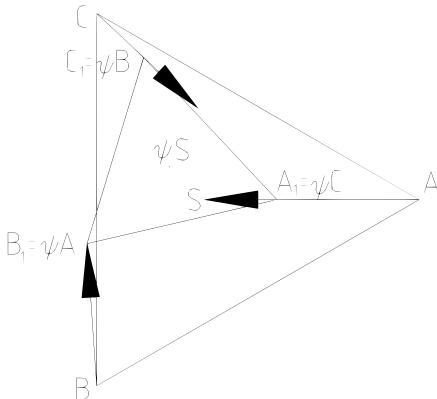


Figure 1. Initial positions, initial vectors, positions after one scheduler's step

We set initial vector \vec{a} to zero. According to the Theorem 3, robot A has to choose S as its destination point, so the vector of movement \vec{a} becomes $\vec{a} = S - A = -1 + 0i$. We program one step of the scheduler to let robot B move to its destination $B_1 = B + \vec{b}$ and to pause robots' A, C movement at the points $A_1 = A + k\vec{a}$, $C_1 = C + l\vec{c}$; $k, l \in (0, 1)$. We are going to find such initial vectors \vec{b}, \vec{c} and constants k, l that the configuration after one scheduler's step will be similar to the starting configuration. The similar projection (we mark it as ψ) must fulfill following equations (see Fig.1):

$$\begin{aligned}\psi A &= B_1 \\ \psi B &= C_1 \\ \psi C &= A_1\end{aligned}$$

Having the projection ψ , all other parameters are just a function of it:

$$\begin{aligned}\vec{b} &= \psi A - B \\ \psi \vec{c} &= S - \psi C\end{aligned}$$

$$\begin{aligned}k &= \frac{|\psi C - A|}{|\vec{a}|} \\ l &= \frac{|\psi B - C|}{|\vec{c}|}\end{aligned}$$

Projection ψ keeps the robots in the equilateral triangle situation.

In i -th scheduler's step, the i -th projection of robot A (i.e. $\psi^i A$) is the robot that has just finished its movement, observes the equilateral triangle situation and computes its destination vector towards the center of the current equilateral triangle – $\psi^i S$. Analogously robot $\psi^i B$ is the one that gets to its destination in the next scheduler step and becomes $\psi^{i+1} A$. Robot $\psi^i C$ is the one that will be paused in the next scheduler's step, becomes $\psi^{i+1} B$, and get to its destination in the second scheduler's step, becomes $\psi^{i+2} A$. When we formalize these properties, we get the following equations and restrictions: insolvable

$$\mathcal{I}(\psi C) = 0 \quad (1)$$

$$\mathcal{I}(\psi^2 B) = 0 \quad (2)$$

$$\psi^3 A = 0 + 0i \quad (3)$$

$$\mathcal{R}(\psi C) \in (0, 1) \quad (4)$$

$$\mathcal{R}(\psi^2 B) \in (0, \mathcal{R}(\psi C)) \quad (5)$$

where $\mathcal{I}(x)$ is the imaginary part of the complex number x and $\mathcal{R}(x)$ is its real part.

Equations 1, 2 and Restrictions 4, 5 formalizes the fact, that robot A is in the next two scheduler's steps on its way from point $1 + 0i$ to the point $0 + 0i$. Equations 3 means that robot A gets to the point $0 + 0i$ in the third scheduler's step.

Now, let us define the mentioned projection ψ . It is characterized by two complex numbers \vec{p} (scaling and rotation) and \vec{q} (movement). Let W be any point, $\psi(W) = \vec{p}W + \vec{q}$. Let \vec{w} by any vector, $\psi(\vec{w}) = \vec{p}\vec{w}$.

From Equation 3 we get

$$\psi^3(1 + 0i) = 0 + 0i \Rightarrow \vec{q} = -\frac{\vec{p}^3}{\vec{p}^2 + \vec{p} + 1}$$

Let $\vec{p} = x + yi; x, y \in \mathbb{R}$.

By adjustment of Equation 1 we get

$$\mathcal{I}(\psi C) = \mathcal{I}\left(\vec{p}\left(-\frac{1}{2} + \frac{\sqrt{3}}{2}i\right) - \frac{\vec{p}^3}{\vec{p}^2 + \vec{p} + 1}\right) = 0$$

By further adjustment in real numbers we get:

$$\begin{aligned}\left(\frac{\sqrt{3}}{2}x - \frac{3}{2}y\right)((x^2 - y^2 + x + 1)^2 + (2xy + y)^2) + \\ + 2xy + y = 0\end{aligned}$$

By adjustment of Equation 2 we get

$$\mathcal{I}(\psi^2 B) = \mathcal{I}\left(\vec{p}^2\left(-\frac{1}{2} - \frac{\sqrt{3}}{2}i\right) - \frac{\vec{p}^3(\vec{p} + 1)}{\vec{p}^2 + \vec{p} + 1}\right) = 0$$

By further adjustment in real numbers we get:

$$\begin{aligned}\left(-3xy - \frac{\sqrt{3}}{2}(x^2 - y^2)\right)((x^2 - y^2 + x + 1)^2 + (2xy + y)^2) - \\ - y(x^2 - y^2 + x + 1) + (x + 1)(2xy + y) = 0\end{aligned}$$

In general, our problem can be seen as solving a system of two equations or finding a common root of functions

$$\begin{aligned} f(x, y) &= \left(\frac{\sqrt{3}}{2}x - \frac{3}{2}y \right) \\ &\cdot ((x^2 - y^2 + x + 1)^2 + (2xy + y)^2) + \\ &+ 2xy + y \end{aligned} \quad (6)$$

$$\begin{aligned} g(x, y) &= \left(-3xy - \frac{\sqrt{3}}{2}(x^2 - y^2) \right) \\ &\cdot ((x^2 - y^2 + x + 1)^2 + (2xy + y)^2) - \\ &- y(x^2 - y^2 + x + 1) + (x + 1)(2xy + y) \end{aligned} \quad (7)$$

Not all roots of these functions are correct solutions of our problem (because we ignored the Restrictions 4, 5), but one of the correct approximate solutions we have found by an numeric algorithm is

$$\begin{aligned} \vec{p} &= -0.381503871916 - 0.357088906042i \\ \vec{q} &= -0.162229750616 + 0.1518475917i \\ \vec{b} &= -0.043733622532 + 0.660784089442i \\ \vec{c} &= 0.4719173352574 - 0.441716211537i \\ k &= 0.6622297506162 \\ l &= 0.4646451929566 \end{aligned}$$

Simulation of this solution is pictured in Figure 2. See appendix A for the proof that there exists an exact solution near the approximate one.

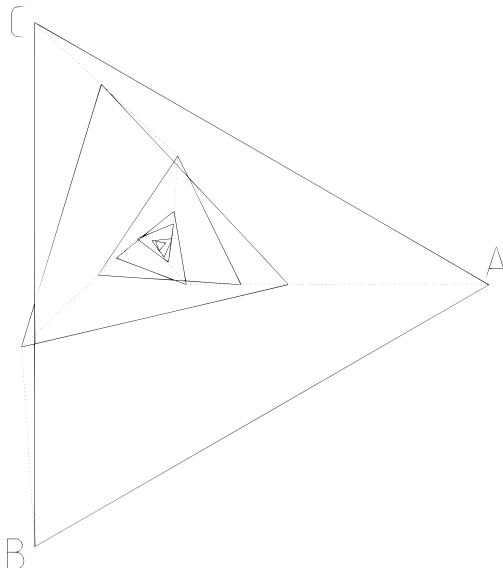


Figure 2. Simulation of the solution

Theorem 4 *The gathering problem is unsolvable for three robots in the model with initial vectors even with the multiplicity detection capability.*

If the robots' algorithm does not choose for the equilateral triangle observation the center of that triangle, we use the Theorem 3. In the

other case we have constructed the initial robots' positions, initial robots' vectors and the scheduler such that after three scheduler's steps we get the configuration similar (projection ψ^3) to the initial one (robots positions are projected too).

If we relax the never mentioned assumption, that robots are on distinct positions at the start, we easily get the following corollary.

Theorem 5 *The gathering problem is unsolvable for $3k$ robots in the model with initial vectors even with the multiplicity detection capability.*

Repeat the proof of the Theorem 4 with k robots starting at point A , k robots at point B and k robots at point C . Scheduler will schedule all robots starting at one point totally synchronously and thus robots will be moving together and do not get any more valuable input than in the original proof.

Theorem 6 *Basic model with multiplicity detection capability and the model with initial vectors and multiplicity detection capability are different.*

Gathering problem is solvable for at least 3 robots in the basic model with multiplicity detection capability and is unsolvable for 3 robots in the model with initial vectors even with multiplicity detection capability.

6 CONCLUSIONS AND OPEN PROBLEMS

We have introduced the model of asynchronous mobile robots with initial vectors and showed some of its basic properties. Then we proved that the gathering problem is unsolvable for three robots in this model even with the multiplicity detection capability and thus that the basic model with multiplicity detection and the model with initial vectors and multiplicity detection are different.

The gathering problem remains open in the model with initial vectors and the multiplicity detection for more than three robots.

Other problems studied in the basic model with zero initial vectors can be investigated in the introduced model with nonzero initial vectors.

ACKNOWLEDGEMENTS

Thanks Authors would like to thank Richard Královič, Rastislav Královič and Dana Pardubská for their support.

REFERENCES

- [1] N. Agmon and D. Peleg. Fault Tolerant Gathering Algorithms for Autonomous Mobile Robots. In *Proc. 15th Symposium on Discrete Algorithms (SODA 2004)*, pages 1063-1071, 2004.
- [2] M. Cieliebak, P. Flocchini, G. Prencipe and N. Santoro. Solving the Gathering Problem. In *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP '03)*, pages 1181-1196, 2003.
- [3] M. Cieliebak and G. Prencipe. Gathering Autonomous Mobile Robots. In *Proc. of 9th International Colloquium On Structural Information And Communication Complexity (SIROCCO 9)*, pages 57-72, 2002.
- [4] R. Cohen and D. Peleg. Convergence Properties of the Gravitational Algorithm in Asynchronous Robot Systems. *SIAM Journal of Computing*, 34(6): 1516-1528, 2005.
- [5] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Asynchronous Mobile Robots with Limited Visibility. In *Proc. 18th Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, pages 247-258, 2001.

- [6] P. Flocchini, G. Prencipe, N. Santoro and P. Widmayer. Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. In *Proc. 10th Annual International Symposium on Algorithms and Computation (ISAAC '99)*, pages 93–102, 1999.
- [7] B. Katreňák. Biangular Circle Formation by Asynchronous Mobile Robots. In *Proc. of 12th International Colloquium On Structural Information And Communication Complexity (SIROCCO 12)*, pages 185–199, 2005.
- [8] G. Prencipe. On the Feasibility of Gathering by Autonomous Mobile Robots. In *Proc. of 12th International Colloquium On Structural Information And Communication Complexity (SIROCCO 12)*, pages 246–261, 2005.
- [9] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM Journal of Computing*, 28(4):1347–1363, 1999.

A Existence of Exact Solution

One of the approximate solutions of the Equations 6, 7 is

$$p = x + yi = -0,381503871916 - 0,357088906042i$$

We are going to prove that there exist an exact solution in its surrounding – in interval $\langle -0.382, -0.381 \rangle \times \langle -0.358, -0.356 \rangle$.

In the beginning we mention some assertions from function analysis, which we use later in the proof.

Theorem 7 Let $h : \langle a, b \rangle \rightarrow \mathbf{R}$ be a continuous function. If h is decreasing (resp. increasing) on $\langle a, b \rangle$ and $h(a) \cdot h(b) < 0$, then exists exactly one $c \in \langle a, b \rangle$, such that $h(c) = 0$.

Theorem 8 Let $h_1, h_2 : \langle a, b \rangle \rightarrow \mathbf{R}$ be continuous functions. If $h_1(a) < h_2(a)$ and $h_1(b) > h_2(b)$ then there is $c \in \langle a, b \rangle$ such that $h_1(c) = h_2(c)$.

Now we are able to prove the main theorem. In further text, we denote $x_1 = -0.382, x_2 = -0.381, y_1 = -0.358, y_2 = -0.356, y_0 = -0.357$ and $J = \langle x_1, x_2 \rangle, I = \langle y_1, y_2 \rangle$. Refer to the Figure 3 for the depicted behavior of functions $f(x, y)$ and $g(x, y)$ on the interval $J \times I$.

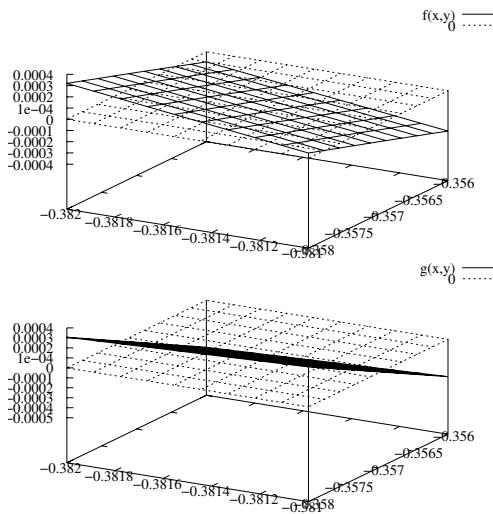


Figure 3. Functions $f(x, y)$ and $g(x, y)$ near the approximate solution

Theorem 9 Let $f(x, y)$ and $g(x, y)$ be functions as defined earlier in Equations (6), (7). There exists $x \in \langle x_1, x_2 \rangle, y \in \langle y_1, y_2 \rangle$ such that $f(x, y) = g(x, y) = 0$.

After substituting in functions f and g we get

- $f(x_1, y_0) > 0$ and $f(x_1, y_2) < 0$ and from Mean-value Theorem there exists $y'_f \in (y_0, y_2)$ such that $f(x_1, y'_f) = 0$
- $f(x_2, y_1) > 0$ and $f(x_2, y_0) < 0$ and from Mean-value Theorem there exists $y''_f \in (y_1, y_0)$ such that $f(x_2, y''_f) = 0$
- $g(x_1, y_1) > 0$ and $g(x_1, y_0) < 0$ and from Mean-value Theorem there exists $y'_g \in (y_1, y_0)$ such that $g(x_1, y'_g) = 0$
- $g(x_2, y_0) > 0$ and $g(x_2, y_2) < 0$ and from Mean-value Theorem there exists $y''_g \in (y_0, y_2)$ such that $g(x_2, y''_g) = 0$

Let define functions on interval $\langle x_1, x_2 \rangle$ as $h_f(x) = y \iff f(x, y) = 0$ and $h_g(x) = y \iff g(x, y) = 0$. Since the function f is decreasing on $\langle y_1, y_2 \rangle$ for every fixed $x \in \langle x_1, x_2 \rangle$ (proved later in Theorem 10), from Theorem 7 for every x there exists exactly one $y \in \langle y_1, y_2 \rangle$ such that $f(x, y) = 0$. Thus the function h_f is well defined. The same holds for h_g (since g is decreasing on $\langle y_1, y_2 \rangle$, too). Moreover both h_f and h_g are continuous, since they are product of intersection of continuous function f (resp. g) and a plane.

Since easily holds $y'_f = h_f(x_1) > h_g(x_1) = y'_g$ and $y''_f = h_f(x_2) > h_g(x_2) = y''_g$ and both h_f and h_g are continuous, from Theorem 8 there exists $x \in (x_1, x_2)$ and $y \in (y_1, y_2)$ such that $h_f(x) = h_g(x) = y$ i.e. $f(x, y) = g(x, y) = 0$.

Theorem 10 The function $f(x, y)$ is on interval I decreasing for every fixed $x \in J$.

The function $f(x, y)$ on the interval I is decreasing for every fixed x iff $\frac{\partial f(x, y)}{\partial y}$ is defined and negative for every $y \in I$.

The function $\frac{\partial f(x, y)}{\partial y}$ is defined for every $(x, y) \in J \times I$ and since $y_1 \leq y \leq y_2$ and $x_1 \leq x \leq x_2$, the inequality

$$\begin{aligned} -7.5y^4 &+ 2\sqrt{3}xy^3 &- 9x^2y^2 &- 9xy^2 &+ \\ &+ 4.5y^2 &+ 2\sqrt{3}x^3y &- \sqrt{3}x^2y &- \\ &- \sqrt{3}xy &+ 2y &+ 1.5x^4 &- \\ &- 3x^3 &- 4.5x^2 &- x - 0.5 &\leq \\ -7.5y_2^4 &+ 2\sqrt{3}x_1y_1^3 &- 9x_2^2y_2^2 &- 9x_1y_1^2 &+ \\ &+ 4.5y_1^2 &+ 2\sqrt{3}x_1^3y_1 &- \sqrt{3}x_1^2y_1 &- \\ &- \sqrt{3}x_2y_2 &+ 2y_2 &+ 1.5x_1^4 &- \\ &- 3x_1^3 &- 4.5x_2^2 &- x_1 - 0.5 &\doteq \\ \doteq -0.349586 & \text{ holds and } \frac{\partial f(x, y)}{\partial y} < 0 \text{ for each } (x, y) \in J \times I. \end{aligned}$$

Testing the limits of emergent behavior in MAS using learning of cooperative behavior

Jordan Kidney and Jörg Denzinger¹

Abstract. We present a method to test a group of agents for (unwanted) emergent behavior by using techniques from learning of cooperative behavior. The general idea is to mimick users or other systems interacting with the tested agents by a group of tester agents and to evolve the actions of these tester agents. The goal of this evolutionary learning is to get the tested agents to exhibit the (unwanted) emergent behavior. We used our method to test the emergent properties of a team of agents written by students for an assignment in a basic MAS class. Our method produced tester agents that helped the tested agents to perform at their best and another configuration of our system showed how much the tested agents could hold their own against very competitive agents, which revealed breakdowns in the tested agents' cooperation.

1 Introduction

Emergent behavior has become a hot research topic in both the multi-agent systems (MAS) and the artificial life communities. Having a group of agents that –by working together– achieve goals that are beyond what can be expected from them by just adding up their individual abilities is naturally a very desirable goal (especially given that the computer “world” is getting more and more distributed), so that even researchers from areas like software engineering and computer security have become interested in this phenomenon. While there are quite a few examples for the synergetic effects that emergent behavior can achieve (see, for example, [2]), there are still many aspects of emergent behavior that are not well understood. Current research mostly focuses on finding concepts for agent cooperation or interaction that result in some intended emergent behavior.

An aspect of emergent behavior that has not been addressed very much, so far, is the fact that the interactions of a set of agents might not always produce only wanted emergent behavior (which is naturally known, but not much written of) and how to identify situations for a set of agents when unwanted behavior surfaces. Traditionally, finding unwanted behavior of a system is the task of testing the system. But software testing in general struggles to keep up with the growing complexity of todays software systems and there have been very few approaches to deal with testing of multi-agent systems beyond the standard techniques developed decades ago. Following test case scripts, testing of individual system components and visualization of agent interactions help with testing of multi-agent systems, but finding unwanted emergent behavior of these systems requires much more than these techniques. The growing number of cases of users abusing systems by providing unexpected inputs or exploiting bugs in the systems is a clear proof of that (see [13]).

In this work, we propose to tackle this problem by making use of techniques that helped creating the problem in the first place: *learning of cooperative behavior* of agents. The general idea is to employ a group of cooperating agents that interact with the multi-agent system that we want to test in order to get this multi-agent system to produce an unwanted (emergent) behavior. Learning methods are used by the group of tester agents to adapt their behavior based upon observations made about the tested system. This process is done to get the tested system closer and closer to the unwanted behavior we are looking for. More precisely, we propose to use ideas from evolutionary learning of behavior (see [3]) to develop a set of tester agent strategies that forces the tested system to show the unwanted behavior we are looking for.

We used our method to test the emergent behavior of a set of agents created by students for their assignment in our multi-agent systems course. The tested agents act within the ARES 2 simulator (see [4]) that is a disaster simulation in which the tested agents have to find and rescue survivors buried under rubble. With our testing method we are able to evaluate the limits of the emergent behavior of the student teams. This testing on one hand works at creating a team of tester agents that tries to help the tested agents as much as possible, boosting the number of survivors rescued by the tested agents. On the other side it works to also create tester agents that keep the score of the tested agents as low as possible, revealing weaknesses in the cooperation strategies employed by the students. Our experimental evaluation shows that our testing method allows us to reveal quite well the (positive and negative) limits of the tested student team to help us in our evaluation (and grading) of the tested team. Additionally, our agents revealed confusion in the behavior of the tested agents in some circumstances, clearly unwanted emergent behavior.

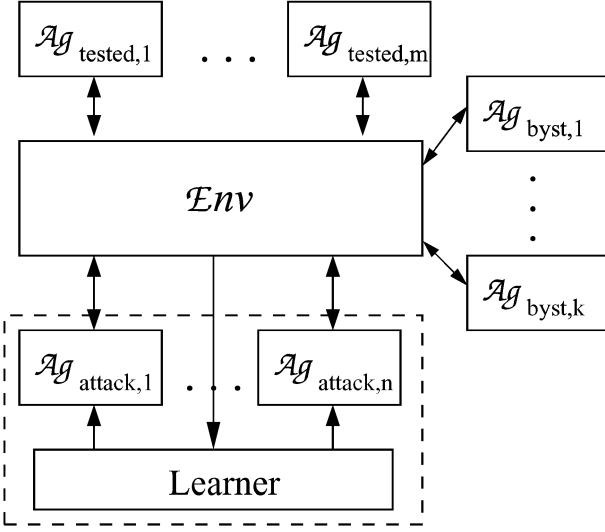
2 Basic definitions

A general definition that can be instantiated to most of the views of agents in literature sees an agent Ag as a quadruple $Ag = (Sit, Act, Dat, f_{Ag})$. Sit is a set of situations the agent can be in, Act is the set of actions that Ag can perform and Dat is the set of possible values that Ag 's internal data areas can have. In order to determine its next action, Ag uses $f_{Ag} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas.

In order to interact, a group of agents has to share an environment or at least parts of it. Therefore, a multi-agent system MAS on a very high level is a pair $MAS = (A, Env)$, with A a set of agents and Env the environment. Env consists of a set of environment states. Based on this view of a multi-agent system, the first 3 components of an agent can be more structured by distinguishing between parts dealing with other agents and parts dealing with the

¹ Department of Computer Science, University of Calgary, Canada email: {kidney,denzinge}@cpsc.ucalgary.ca

Figure 1. General setting of our approach



agent in the environment, but we do not need this distinction in this paper. Note that communication between agents is modeled by having on the one side actions and on the other side the communication “channels” provided by the environment. Note also that the set Sit of an agent can be just a view on the environment, i.e. an element of Sit can contain less information than an element of $\mathcal{E}nv$.

3 Testing by evolutionary learning

The method we propose for testing for unwanted emergent behavior of a group of agents A_{tested} uses a set of (learning) tester (or attack²) agents A_{attack} that interact with the agents in A_{tested} by playing the role of entities that the agents in A_{tested} usually interact with in their environment. The agents in A_{attack} can play the role of “team members” of the agents in A_{tested} or the role of users or other systems that use/interact with the agents in A_{tested} . Depending on the circumstances, there can be a third set A_{byst} of agents, the bystander agents, which are agents that are not part of the set of tested agents, but interact with them in the environment, and are also not under control of the tester. Figure 1 presents a view of the general setting just described, i.e. all 3 types of agents act in the environment $\mathcal{E}nv$ and by doing that they interact with each other.

Figure 1 also indicates that the agents in A_{attack} are learning, but how learning is organized is indicated rather vague, we only see that the learning gets feedback by observing the environment. There are many different ways how learning of a behavior for a group of agents can be accomplished (see [8] for an overview). We can have one learner that learns off-line and creates a team of attack agents that are not doing any learning while they “work”. The other extreme is that each agent $Ag_{attack,i}$ in A_{attack} has as part of it a learning component and the agent is learning on-line. And, as pointed out in [5], there are many other possibilities in between those extremes. We think that different combinations of A_{tested} and $\mathcal{E}nv$ will require different learning methods for A_{attack} .

In this paper, we propose a central off-line learning approach, i.e. the first extreme mentioned above. Naturally, learning is strongly connected with the agent architecture (Sit, Act, Dat, f_{Ag}) of an agent

² We use the term attack in the following, since “tester” and “tested” are very difficult to distinguish

and what architecture to choose depends a lot on the task agents have to perform. From the point of view of testing the limits of emergent behavior of agents in A_{tested} , the task of the agents in A_{attack} is to provoke the agents in A_{tested} into showing a particular behavior (that the human tester would like to know about).

More precisely, from the point of view of an outside observer, each agent $Ag_{attack,i} \in A_{attack}$ has to perform a sequence of actions $a_{i,1}, \dots, a_{i,l_i}$ ($a_{i,j} \in Act_{attack,i}$) that results in a sequence e_0, e_1, \dots, e_x of (enhanced) environment states, such that some condition $\mathcal{G}_{emerge}((e_0, e_1, \dots, e_x))$ is true. By an (enhanced) environment state, we mean a view on $\mathcal{E}nv \times \bigcup_{Ag \in A_{tested}} Dat_{Ag}$, i.e. we allow on the one side that internal data of the tested agents can be made available to our learning agents and on the other side we might not even make everything that is observable in the environment available to our attack agents (and the learner for them). If all agents interact with the environment in a synchronous fashion, then the length of the action sequences of the attack agents is the same and this length is also the number of environment states that we observe after starting from e_0 , i.e. $l_i = l_j$ and $l_i = x$ for all i, j .

In the case of the agents interacting asynchronously, we need to record the (enhanced) environmental state every time a change takes place. Then x obviously cannot easily be connected to the l_i s. In the asynchronous case, it often can be necessary to add timing information to the action sequence of an $Ag_{attack,i}$, resulting in a sequence $(t_{i,1}, a_{i,1}), \dots, (t_{i,l_i}, a_{i,l_i})$ of timed actions, with the $t_{i,j}$ indicating time units and $a_{i,j} \in Act_{attack,i}$ as before. Then $Ag_{attack,i}$ lets $t_{i,j}$ time units pass between performing $a_{i,j-1}$ and $a_{i,j}$.

Based on this action sequence oriented view on the agents in A_{attack} , the learning task for these agents is to come up with an action sequence for each agent that results in a sequence of environment states fulfilling \mathcal{G}_{emerge} . Evolutionary learning methods have proven to be quite successful for such settings (see [3], [1]). An evolutionary algorithm works on a set of so-called individuals (the set being called a population). The individuals in a population are evaluated using a fitness measure fit and so-called genetic operators are applied to selected individuals to create new individuals. The selection process usually contains both a fitness-oriented and a random component. The new individuals replace the worst fit individuals in the old population and this process is repeated until an end condition is fulfilled.

We instantiate this very general scheme in the following manner to learn the action sequences for the agents in A_{attack} . An individual consists of an action sequence $a_{i,1}, \dots, a_{i,l_i}$ for each of the agents $Ag_{attack,i} \in A_{attack}$ (or $(t_{i,1}, a_{i,1}), \dots, (t_{i,l_i}, a_{i,l_i})$ if we have an asynchronous system). The fitness of an individual $((a_{1,1}, \dots, a_{1,l_1}), \dots, (a_{n,1}, \dots, a_{n,l_n}))$ is based on evaluating the success of the individual when applied as the strategy for the agents in A_{attack} in one or several runs. This means we feed the action sequences to the agents they are intended for, let these agents interact with the other agents, i.e. the agents in A_{tested} and A_{byst} , and the environment and produce the sequence e_0, e_1, \dots, e_x of (enhanced) environment states that is the consequence of executing the individual. Due to having bystanders, an individual might not produce the same environment states every time the attack agents use this individual (other reasons for this can be random elements in the tested agents or random events occurring in the environment). If this is the case, we perform several runs to evaluate an individual and sum up the fitness of these runs.

To evaluate a run of an individual, we want to measure how near this run came to fulfilling the condition \mathcal{G}_{emerge} . Obviously, this depends on \mathcal{G}_{emerge} and on what information is represented in an en-

hanced environment state. But we found it very useful to base the fitness of a single run *single_fit* on an evaluation of the environment state sequence after every new state (see also [1]). Thus we have

$$\text{single_fit}((e_0, \dots, e_x)) = \begin{cases} j, & \text{if } \mathcal{G}_{\text{emerge}}((e_0, \dots, e_j)) = \text{true} \\ & \text{and } \mathcal{G}_{\text{emerge}}((e_0, \dots, e_i)) \\ & = \text{false for all } i < j \\ \sum_{i=1}^x \text{near_emerge}((e_0, \dots, e_i)), & \text{else.} \end{cases}$$

As the name *near_emerge* suggests, this function is supposed to measure how near its argument state sequence comes to fulfilling the predicate $\mathcal{G}_{\text{emerge}}$ (so, small *near_emerge* values indicate good individuals). We will provide an example for how to create such a function for an application in Section 4.2.

There are several schemata for genetic operators that can be used for creating new individuals out of individuals of the form we are using. Here we present four such schemata that we used in our application. Our *standard crossover* needs two individuals $((a_{1,1}, \dots, a_{1,l_1}), \dots, (a_{n,1}, \dots, a_{n,l_n}))$ and $((b_{1,1}, \dots, b_{1,l_1}), \dots, (b_{n,1}, \dots, b_{n,l_n}))$. The operator selects randomly a position j in the action sequence of all agents and performs then a standard crossover on each of the strings. The result is then $((a_{1,1}, \dots, a_{1,j}, b_{1,j+1}, \dots, b_{1,l_1}), \dots, (a_{n,1}, \dots, a_{n,j}, b_{n,j+1}, \dots, b_{n,l_n}))$.

A *standard mutation* also selects a position j in one individual randomly and changes the action there of every agent to another one, resulting in $((a_{1,1}, \dots, a_{1,j-1}, a'_{1,j}, a_{1,j+1}, \dots, a_{1,l_1}), \dots, (a_{n,1}, \dots, a_{n,j-1}, a'_{n,j}, a_{n,j+1}, \dots, a_{n,l_n}))$ with $a'_{i,j} \in \text{Act}_{\text{attack},i}$ and $a_{i,j} \neq a'_{i,j}$.

We also have so-called targeted genetic operators. Instead of selecting a random position in an action sequence, these operators make use of the fitness of the parent individuals to determine “good” positions. More precisely, we look at the *near_emerge* values for all the starting state sequences and identify positions, where taking in the next state into the sequence results in a substantially higher *near_emerge* value for this new sequence (indicating that something happened that leads us away from our goal). If we change the actions of our attack agents before this happens, then there is a good chance that we can avoid this. So, for each individual we remember the first index j , such that

$$\text{near_emerge}((e_0, \dots, e_j)) \geq \text{near_emerge}((e_0, \dots, e_{j-1})) + \text{too_much_lost},$$

for a parameter *too_much_lost*. *Targeted crossover* then first identifies the agent indexes with actions in their sequences that are leading to an environment state between $e_{j-\text{cotarg}}$ and e_j (with *cotarg* being another parameter). Targeted crossover then performs a standard crossover, but selects the agent index and the action sequence position only out of the ones identified in the first step. *Targeted mutation* does the same using standard mutations (with a parameter *muttarg*).

4 Experimental evaluation: testing student teams in ARES 2

In this section, we present our experiences in applying our concept from the last section to testing a student team written for the ARES 2 simulator. Our concept allowed us on the one side to test how well this team can make use of agents that try everything to help the team, while we also used the concept to create teams of attack agents that tried to achieve a very bad performance of the tested agent team.

4.1 ARES 2

ARES is a system that simulates a city struck by an earthquake and rescue robots that work together to locate and dig out survivors of the earthquake. In contrast to the RoboCupRescue simulator (see [10]) that provides the same general scenario, the goal in developing ARES was not to be as realistic as possible to allow the development of robot controls that can be used in the real world, the goal was to provide abstractions and simplifications of the real world that allow a team of students in an introductory class on multi-agent systems to enhance their learning experience by creating controls for the robots/agents that rescue many survivors in various problem scenarios within ARES (see [4]). The last version of ARES, ARES 2, allows for the interaction of several student teams within one simulation, thus forcing the students to think about how their agents should interact with other rescue agents, allowing for many possibilities for emergent behavior.

Since ARES was developed as a teaching tool, it contains various parameters, so-called world rules, that produce rather different settings for the students (each semester, a different combination of world rule instantiations can be used to avoid too much “inspiration” from the last semester). For example, the two instantiations for the world rule that determines how agents can replenish their energy can either allow the students to mainly neglect planning around energy (if agents can simply sleep everywhere) or planning around resources becomes very important (if agents can only recharge at certain places in the world). As a consequence, instructors using ARES often do not know what the best concepts for a particular instantiation of the world rules are, which makes evaluating the teams of the students difficult. Especially evaluating how the students deal with other teams is very difficult, since using fixed teams that interact with a student team allows the students to optimize towards these teams instead of developing generally good concepts. Therefore this situation is a perfect test application for our concept for testing emergent behavior.

A simulation run in ARES consists of a sequence of rounds. In each round, ARES allows every rescue agent to submit a non-communication action and, depending on the world rules, either alternatively a communication action or one or several additional communication actions. Each agent is allotted a certain amount of CPU-time each round for making its decisions. ARES then computes out of all action requests the new world state and sends to each agent its new world view. An ARES world (or scenario) consists of a collection of connected grids and each grid contains a so-called stack, where each stack element is either a piece of rubble, requiring a certain number of agents to be removed, or a survivor or survivor group. Associated with each grid field is a move cost indicating the energy an agent loses when moving on the grid. In addition to standard grids, there are fire and instant death grids that “kill” an agent moving on them and there are also recharge grids.

Agents can move from grid to grid, observe the world, remove rubble by digging and rescue survivors. Additionally, agents can send messages to other agents in their team, allowing for planning and cooperation. In order to remove a piece of rubble, this piece has to be on top of the stack of a grid, at least the required number of agents has to perform digging in the same round and, naturally, these agents have to be located on the grid at that time. The agents can belong to different teams. To rescue survivors, they have to be on top of the stack and at least one agent has to perform the rescue survivor action. A world rule allows for different ways of scoring for rescuing survivors if several agents perform the rescue at the same time.

In addition to updating the world and doing the scoring, ARES

also keeps track of the energy levels of survivors and agents, letting them die if the level gets to zero or below. ARES also delivers the messages agents send to each other in the round after the round in which the message was sent.

4.2 Instantiating our method

The interaction of agents within ARES is organized in rounds, which means that we are dealing with the synchronous case of Section 3. For each scenario in ARES, there is a given number of rounds l in the simulation, which is then also the length of the action sequences of the agents in A_{attack} and the sequence of environment states. Obviously, ARES represents the environment \mathcal{E}_{nv} and the agents of the students are the agents in A_{tested} . In our experiments, the set A_{byst} was empty, but if we used a second student team, these agents could be the set A_{byst} .

While the agents in A_{tested} naturally used all the actions available to agents in ARES, our attack agents do not need to use any communication actions since their coordination is achieved by the learning process. An environment state is a state of ARES, i.e. the state of the world (all grids and their stacks, positions of all agents, energy levels of survivors and agents) and the scores of all teams. Naturally, we did not want to change the agents of the students to get internal information out of them. Therefore we did not require to enhance the environment states.

To evaluate the agent team of the students, we would like to know how good this team is if the other team, i.e. the agents in A_{attack} , tries to provide help and if this other team tries to actively work against the student team. This means that there is not exactly a particular behavior we are looking for that is shown or not. Instead of having a predicate G_{emerge} , we have a function f_{emerge} that on the one side we want to maximize and on the other side to minimize and this function just counts the survivors the student team got points for. So, within our general scheme of Section 3, G_{emerge} is always false and f_{emerge} will contribute to our function near_emerge .

Our near_emerge function that aims at producing agents in A_{attack} that boost the performance of the agents in A_{tested} is based on the ideas of having the agents in A_{attack} stay near the agents of A_{tested} , so that they can help dig, and of trying to get a high f_{emerge} value for the student team. More precisely, we have

$$\text{near_emerge}_{\text{pos}}((e_0, \dots, e_j)) = \sum_{i=1}^n \text{dist}(A_{g_{\text{attack}}, i}, e_j) - 2 * f_{\text{emerge}}(e_j)$$

where $\text{dist}(A_{g_{\text{attack}}, i}, e_j)$ is the distance between $A_{g_{\text{attack}}, i}$ and the closest agent from A_{tested} in state e_j .

For producing attack teams that show off the possible negative behavior of the student team, we just use the f_{emerge} value, i.e.

$$\text{near_emerge}_{\text{neg}}((e_0, \dots, e_j)) = f_{\text{emerge}}(e_j)$$

In our experiments, this simple function was already sufficient. Note that our algorithm tries to minimize the fitness, so that large near_emerge values make a strategy for A_{attack} bad.

We would like to point out that this instantiation of our general concept is independent of all the world rules in ARES, so that it can be used for all settings of these rules.

4.3 Experiments

For our experiments, we followed the advice in [4] and created two types of scenarios for ARES, the student team and our system: special scenarios that aim at evaluating special aspects of the students'

team and random scenarios that come nearer to what a real-live scenario usually might be. In each scenario/world we have 2 agents from the student team and 2 agents in A_{attack} . The time given to the teams is always 50 rounds and each team participating in a save gets the points for this save. Each agent is given 1 second per round to do its "thinking", which means that a single simulation run takes roughly 200 seconds.

The settings for the evolutionary learner were as follows: We used a very small population size of 7 and had 10 generations only, to account for the rather time consuming evaluation of each individual. Nevertheless, this still means that a complete run of our system (i.e. computing the entry for pos or neg in one entry of Table 1) takes around 5 hours. The selection of the parents for the genetic operators was done using a 3-tournament selection, i.e. each time we need a parent, we randomly select 3 individuals from the population and then select out of these 3 the one with the best (lowest) fitness value. *too_much_lost* was chosen so that a scored point for the tested team would create a targeted position (for the neg case). Then the other parameters for the targeted genetic operators were set so that the position in the sequence used by the operator was directly before the drop in *near_emerge* occurred.

Table 1. Points scored by A_{tested}

World	1×st	2×st	pos	neg
Spec1	6	5	7	2
Spec2	4	8	8	4
Spec3	8	6	9	1
Rand1	11	8	12	6
Rand2	12	9	13	8
Rand3	14	11	16	10

In the 5×5 world Spec1 we have 7 survivors that are buried under rubble, the ones at (1,3), (5,3) and (3,5) (with (1,1) being the upper left corner of the world) requiring 2 agents to dig them out, the ones at (1,1) and (5,1) needing 3 agents and the ones at (1,5) and (5,5) just one. Additionally, at (3,3) there is an unburied survivor. The 2 agents of the student team are at (3,1) and the attack agents flank the tested agents at (1,2) and (5,2). The world Spec2 is a 7×7 world with a survivor in each corner that needs 2 agents to be dug out, 6 survivors placed into the corners of the 3×3 square in the center of the world with 2 survivors in each of the left corners and 1 survivor in each of the right corners, a survivor requiring 3 agents in the center of the world, a survivor requiring 4 agents 2 grid fields below this center and one survivor requiring 3 agents two fields above the center. One tester and one attack agent start at (1,4) and the other two agents start at (7,4). Spec3 is another 5×5 world with buried survivors requiring 2 agents at (2,2), (3,2), (4,2), (2,4), (3,4) and (4,4), a survivor requiring 3 agents in the center of the world and a survivor in the open at (1,3) and (5,3). One agent from each team starts at (3,1) and (3,5).

The worlds Rand1, Rand2, and Rand3 were generated using ARES 2's functionality for creating random worlds. All 3 worlds are 7×7 and require at a maximum 3 agents to rescue a survivor (and naturally not all of the time so many). In contrast to the special worlds, different grid fields can have different move costs. The start positions for the agents were random, but we required that always one tested agent and one attack agent started together at the same grid. All random worlds had 99 survivors in them, with random amounts of energy to start with.

In Table 1, the column $1 \times st$ reports on the success of the students' team when working alone in a world, while column $2 \times st$ gives the results of two incarnations of the students' team. Note that we report in column $2 \times st$ the better score of the two teams. The columns pos, respectively neg report on the scores of the tested team when

working with the best teams our learning system came up with to help the team (pos) and to hinder the team (neg). With the exception of Spec2, we see for all worlds that the learner could produce teams that increased, often substantially, the score of the students' team and also with teams that made it really tough for the tested team. Especially for Spec1 and Spec3 the results of the tested team are much less than in all other test series. The difference for the random worlds is not so big, because in these worlds there are many possibilities for the tested agents, too many to block them all.

While from an instructor's perspective, the information in Table 1 is already very useful for grading different students' teams, because it shows the spectrum a particular team can achieve, a closer look at the runs of the evolved attack teams producing the highlights in this table will show the usefulness of our approach for finding unwanted emergent behavior. Looking at the ARES 2 runs of the final attack teams for column pos, we can not report any surprising things. The attack agents stayed within close range of the tested agents, although they did not always occupy the same grid field. The attack agents learned to dig whenever the tested agents expected them to do this and as a result we got the good scores for the tested agents.

Of more interest are the ARES 2 runs that the attack agents for the neg case for Spec1 and Spec3 produce. Naturally, the attack agents did not do anything that helped the tested agents to get points, even if this meant that they also would not get points. While for the other scenarios the students' strategy for their team was able to realize that they do not get cooperation and from there on the tested agents planned only relying on themselves, the attack teams for Spec1 and Spec3 managed to get the tested agents into different kinds of weird behavior where the tested agents did not even try to dig out survivors (that they could dig out alone) when an attack agent came near and where the tested agents in fact stopped doing anything after a while although that had sufficient energy and time to rescue some more survivors. Essentially, our approach was able to produce evidence for problems in the tested agent design, evidence for clearly unwanted emergent behavior.

5 Related work

The current practice for testing for unwanted behavior in software testing is the use of random interaction sequences (see [6]), followed by human analysis and additional constructed sequences. Obviously, random sequences are just the starting point for our method. The efforts to accommodate the needs of multi-agent systems in testing, so far, either focus on developing graphical visualization tools that allow human testers to better observe the behaviors of the tested agents (see, for example, [7]) or develop special protocols and languages that provide a standard way for a developer to inspect the interactions of agents using them (see, for example, [9]).

The use of evolutionary methods to help with software testing has been suggested by several authors. For example, in [11] known (previous) errors of a system were slightly modified (and combined) by an evolutionary algorithm to create additional test cases around known problematic ones. In [12], evolutionary methods were used to develop tests covering the possible paths through a program. But, to our knowledge, there have been no works that deal with using any artificial intelligence concepts to detect unwanted emergent behavior.

6 Conclusion and future work

We presented a method to test the limits of emergent behavior in multi-agent systems that makes use of evolutionary methods to learn

the behavior of agents that are interacting with the agents to be tested. The goal of these tester/attack agents is to get the tested agents to exhibit a behavior that we are interested in, for example because we do not want the tested agents to show this behavior. We tested our method by testing a team of rescue agents developed by students to act in the ARES 2 system. For a collection of different rescue scenarios, we were able to evolve specific teams that showed off the ability of the tested student team to make use of very cooperative other agents and the inability of the student team to deal well with agents that work against their efforts in some rescue scenarios.

It should be noted that the evolved agents use a very primitive agent architecture, action sequences, and use implicit cooperation due to the learning process, so that these agents are clearly less complex than the agents that they are testing. This is due to the fact that the tester/attack agents can be much more focused and specialized than the agents they test, since they only have to produce a specific behavior and do not have to cover a lot of use scenarios, which naturally the tested agents have to. Given that the complexity of the systems to test is one of the major problems in today's testing efforts, this feature of our method is very promising!

Naturally, there is a lot of future research that we are interested in. In addition to additional applications, we see a lot of possibilities to improve the learning of the cooperative behavior of the tester/attack agents, especially with regard to building in additional knowledge about the application. This ranges from usage of more pre-defined structure in the action sequences, over more application specific fitness functions to additional genetic operators (for example to deal with the timing information). Also, we expect that more complex systems to test might require a little bit more sophisticated agent architectures.

REFERENCES

- [1] B. Chan, J. Denzinger, D. Gates, K. Loose and J. Buchanan, 'Evolutionary behavior testing of commercial computer games', Proc. CEC 2004, Portland, 2004, pp. 125–132.
- [2] J. Denzinger, 'Knowledge-Based Distributed Search Using Teamwork', Proc. ICMAS-95, San Francisco, 1995, pp. 81–88.
- [3] J. Denzinger and M. Fuchs, 'Experiments in Learning Prototypical Situations for Variants of the Pursuit Game', Proc. ICMAS-96, Kyoto, 1996, pp. 48–55.
- [4] J. Denzinger and J. Kidney, *Teaching Multi-Agent Systems using the ARES Simulator*, Italic e-journal 4(3), 2005.
- [5] J. Denzinger and M. Kordt, 'Evolutionary On-line Learning of Cooperative Behavior with Situation-Action-Pairs', Proc. ICMAS-2000, Boston, 2000, pp. 103–110.
- [6] R.G. Hamlet, 'Predicting dependability by testing', Proc. Intern. Symp. on Software Testing and Analysis, 1996, pp. 84–91.
- [7] L. Lee, D. Ndumu, H. Nwana and J. Collins, 'Visualizing and debugging distributed multi-agent systems', Proc. 3rd AA, 1999, pp. 326–333.
- [8] L. Panait and S. Luke, 'Collaborative Multi-Agent Learning: A Survey', Technical Report GMU-CS-TR-2003-01, Dept. of Comp. Sci. George Mason University, 2003.
- [9] D. Poutakidis, L. Padgham and M. Winikoff, 'Debugging multi-agent systems using design artifacts: The case of interaction protocols', Proc. AAMAS-2002, 2002, pp. 960–967.
- [10] RoboCup Rescue: <http://jelly.cs.kobe-u.ac.jp/robocup-rescue/>. (as viewed on January 3, 2006).
- [11] A.C. Schultz, J.J. Grefenstette and K.A. De Jong, 'Adaptive Testing of Controllers for Autonomous Vehicles', Proc. Symposium on Autonomous Underwater Vehicle Technology, IEEE, 1992, pp. 158–164.
- [12] J. Wegener, 'Evolutionary testing of embedded systems', In *Evolutionary Algorithms for Embedded Systems Design*, Kluwer, 2003, pp. 1–34.
- [13] S. Young and D. Aitel, 'The Hackers Handbook: The strategy behind breaking into and defending networks', Auerbach Publications, 2004.

Boolean games revisited

Elise Bonzon¹ and **Marie-Christine Lagasquie-Schiex¹** and **Jérôme Lang¹** and **Bruno Zanuttini²**

Abstract. Game theory is a widely used formal model for studying strategical interactions between agents. *Boolean games* [8] are two players, zero-sum static games where players' utility functions are binary and described by a single propositional formula, and the strategies available to a player consist of truth assignments to each of a given set of propositional variables (the variables *controlled* by the player.) We generalize the framework to n -players games which are not necessarily zero-sum. We give simple characterizations of Nash equilibria and dominated strategies, and investigate the computational complexity of the related problems.

1 Introduction

Game theory is the most successful formal model for the study of strategical interactions between agents. Informally, a game consists of a set of agents (or players), and for each agent, a set of possible strategies and an utility function mapping every possible combination of strategies to a real value. In this paper we consider only *static* games, where agents choose their strategies in parallel, without observing the others' choices. While game theory considers several formats for specifying a game (especially extended form and normal form, which coincide as far as static games are concerned), they usually consider that utility functions are represented explicitly, by listing the values for each combination of strategies.

In many real-world domains, the strategies available to an agent consist in assigning a value to each of a given set of variables. Now, representing utility functions explicitly leads to a description whose size is exponential both in the number of agents ($n \times 2^n$ values for n agents each with two available strategies) and in the number of variables controlled by the agents ($2 \times 2^p \times 2^p$ values for two agents each controlling p variables). Thus, in many cases explicitly specifying utility functions is unreasonable, as well as computing solution concepts (such as pure-strategy Nash equilibria) using a naive algorithm which enumerates combinations of strategies.

Now, specifying utilities, or more generally preferences, in a compact way, is an issue addressed by many AI researchers in the last years. Languages have been studied which allow for a concise representation of preference relations or utility functions on combinatorial domains, exploiting to a large extent the structural properties of preferences. In this paper, without much loss of generality we focus on the case where each agent has control over a set of propositional (binary) variables. Under this assumption, preferences can be represented within logic-based preference representation languages. Using propositional logic allies a rich expressivity to the possibility of using a wide variety of algorithms.

Since the specification of a static game needs the description of the agents' preferences, it seems natural to specify them using such lan-

guages for compact preference representation. Here, for the sake of simplicity we focus on the simplest possible way of using propositional logic for representing games (the extent to which it can be extended is discussed at the end of the paper), namely by using a revisited form of *Boolean games*. These games [8, 7, 5] are two-players zero-sum games where the players' utilities are binary and specified by a mere propositional formula φ (the *Boolean form* of the game). Some background is given in Section 2. In Section 3, we give a (simplified) description of Boolean games and generalize them so as to represent non zero-sum games with an arbitrary number of players (but we keep the assumption that each player's preferences are represented by a unique propositional formula, inducing a binary utility function). In Sections 4 and 5, we show how well-known tools from propositional logic can be used so as to give simple characterizations of two of the most important game-theoretic notions, namely pure-strategy Nash equilibria and dominated strategies, and so as to derive complexity results for their computation. Sections 6 and 7 respectively address related work and further issues.

2 Background

Let $V = \{a, b, \dots\}$ be a finite set of propositional variables and let L_V be the propositional language built from V and Boolean constants \top (*true*) and \perp (*false*) with the usual connectives. Formulas of L_V are denoted by φ, ψ etc. A *literal* is a variable x of V or the negation of one. A *term* is a consistent conjunction of literals. $\text{Lit}(\alpha)$ denotes the set of literals forming the term α . A formula φ is in DNF when it is written as a disjunction of terms.

2^V is the set of the interpretations for V , with the usual convention that for $M \in 2^V$ and $x \in V$, M gives the value *true* to x if $x \in M$ and *false* otherwise. \models denotes the classical logical consequence relation. Let $X \subseteq V$. 2^X is the set of X -interpretations. A *partial interpretation* (for V) is an X -interpretation for some $X \subseteq V$. Partial interpretations are denoted by listing all variables of X , with a $\bar{\cdot}$ symbol when the variable is set to false: for instance, let $X = \{a, b, d\}$, then the X -interpretation $M = \{a, d\}$ is denoted $\{a, \bar{b}, d\}$. If $\{V_1, \dots, V_p\}$ is a partition of V and $\{M_1, \dots, M_p\}$ are partial interpretations, where $M_i \in 2^{V_i}$, (M_1, \dots, M_p) denotes the interpretation $M_1 \cup \dots \cup M_p$. Let ψ be a propositional formula. A term α is an *implicant* of ψ iff $\alpha \models \psi$ holds. α is a *prime implicant* of ψ iff α is an implicant of ψ and for every implicant α' of ψ , if $\alpha \models \alpha'$ holds, then $\alpha' \models \alpha$ holds. $PI(\psi)$ denotes the set of all the prime implicants of ψ . If $X \subseteq V$, an X -prime implicant of ψ is a prime implicant of ψ such that $\text{Lit}(\alpha) \subseteq X$. $PI_X(\psi)$ denotes the set of all the X -prime implicants of ψ .

Let $\varphi \in L_V$ and $X \subseteq V$. The *forgetting* of X in φ [13], denoted by $\exists X : \varphi$, is defined inductively by: (i) $\exists \emptyset : \varphi = \varphi$; (ii) $\exists \{x\} : \varphi = \varphi_{x \leftarrow \top} \vee \varphi_{x \leftarrow \perp}$; (iii) $\exists (X \cup \{x\}) : \varphi = \exists X : (\exists \{x\} : \varphi)$. Note that $\exists X : \varphi$ is the logically weakest consequence of φ containing only variables from $V \setminus X$ [10, 12].

¹ IRIT, UPS, F-31062 Toulouse Cedex 9, {bonzon, lagasq, lang}@irit.fr

² GREYC, Université de Caen, F-14032 Caen, zanuttini@info.unican.ca

Finally, we denote the partial instantiation of a formula ϕ by an X -interpretation M_X by: $(\phi)_{M_X} = \Phi_{v \in M_X \leftarrow \top, v \in X \setminus M_X \leftarrow \perp}$.

3 Boolean games

Given a set of propositional variables V , a Boolean game on V [8, 7] is a zero-sum game with two players (1 and 2), where the actions available to each player consist in assigning a truth value to each variable in a given subset of V . The utility functions of the two players are represented by a propositional formula ϕ formed upon the variables in V and called *Boolean form* of the game. ϕ represents the goal of player 1: her payoff is 1 when ϕ is satisfied, and 0 otherwise. Since the game is zero-sum³, the goal of player 2 is $\neg\phi$.⁴

Example 1

Consider $V = \{a, b, c\}$. Player 1 controls a and c while 2 controls b . Player 1's goal is $\phi_1 = (a \leftrightarrow b) \vee (\neg a \wedge b \wedge \neg c)$ and therefore, 2's is $\phi_2 = \neg\phi_1 \equiv (\neg a \wedge b \wedge c) \vee (a \wedge \neg b)$. The normal form of this game is depicted on the right (in each (x, y) , x —resp. y —represents the payoff of player 1—resp. 2):

		2	$\{b\}$	$\{\bar{b}\}$
		1	$\{a, c\}$	$\{\bar{a}, c\}$
		2	$\{a, \bar{c}\}$	$\{\bar{a}, \bar{c}\}$
		$\{a, c\}$	(1, 0)	(0, 1)
		$\{a, \bar{c}\}$	(1, 0)	(0, 1)
		$\{\bar{a}, c\}$	(0, 1)	(1, 0)
		$\{\bar{a}, \bar{c}\}$	(1, 0)	(1, 0)

We now give a more general definition of a Boolean game, with any number of players and not necessarily zero-sum (we will see further that the original definition [8, 7] is a special case of this more general framework).

Definition 1 A Boolean game is a 4-tuple (A, V, π, Φ) , where $A = \{1, 2, \dots, n\}$ is a set of players, V is a set of propositional variables (called decision variables), $\pi : A \mapsto V$ is a control assignment function, and $\Phi = \{\phi_1, \dots, \phi_n\}$ is a collection of formulas of L_V .

The control assignment function π maps each player to the variables she controls. For the sake of notation, the set of all the variables controlled by i is written π_i instead of $\pi(i)$. We require that each variable be controlled by one and only one agent, i.e., $\{\pi_1, \dots, \pi_n\}$ forms a partition of V .

Definition 2 Let $G = (A, V, \pi, \Phi)$ be a Boolean game. A strategy s_i for a player i in G is a π_i -interpretation. A strategy profile S for G is an n -tuple $S = (s_1, s_2, \dots, s_n)$ where for all i , $s_i \in 2^{\pi_i}$.

In other words, a strategy for i is a truth assignment for all the variables i controls. Note that since $\{\pi_1, \dots, \pi_n\}$ forms a partition of V , a strategy profile S is an interpretation for V , i.e., $S \in 2^V$.

In the rest of the paper we make use of the following notation, which is standard in game theory. Let $G = (A, V, \pi, \Phi)$ be a Boolean game with $A = \{1, \dots, n\}$, and $S = (s_1, \dots, s_n)$, $S' = (s'_1, \dots, s'_n)$ be two strategy profiles. s_{-i} denotes the projection of S on $A \setminus \{i\}$: $s_{-i} = (s_1, s_2, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$.

Similarly, π_{-i} denotes the set of the variables controlled by all players except i : $\pi_{-i} = V \setminus \pi_i$.

³ Stricto sensu, the obtained games are not zero-sum, but constant-sum (the sum of utilities being 1) – the difference is irrelevant and we use the terminology “zero-sum” nevertheless.

⁴ The original definition [8, 7] is inductive: a Boolean game consists of a finite dynamic game. We use here the equivalent, simpler definition of [5], who showed that this tree-like construction is unnecessary.

Finally, (s_{-i}, s'_i) denotes the strategy profile obtained from S by replacing s_i with s'_i without changing the other strategies: $(s_{-i}, s'_i) = (s_1, s_2, \dots, s_{i-1}, s'_i, s_{i+1}, \dots, s_n)$.

Players' utilities in Boolean games are binary: player i is satisfied by a strategy profile (and gets utility 1) if and only if her goal ϕ_i is satisfied, and she gets utility 0 otherwise. Therefore, the goals $\{\phi_i, i = 1, \dots, n\}$ play the role of the utility functions u_1, \dots, u_n .

Definition 3 For every player i and strategy profile S : $u_i(S) = 0$ if $S \models \neg\phi_i$ and $u_i(S) = 1$ if $S \models \phi_i$.

Example 2 We consider here a Boolean n -players version of the well-known prisoners' dilemma. n prisoners (denoted by $1, \dots, n$) are kept in separate cells. The same proposal is made to each of them: “Either you denounce your accomplices (denoted by D_i , $i = 1, \dots, n$) or you cover them (C_i , $i = 1, \dots, n$). Denouncing makes you freed while your partners will be sent to prison (except those who denounced you as well; these ones will also be freed). But if none of you chooses to denounce, everyone will be freed.”

Here is the representation of this game in normal form for $n = 3$:

		strategy of 3: C_3		strategy of 3: D_3			
		2	C_2	D_2	C_2	D_2	
		1	C_1	$(1, 1, 1)$	$(0, 1, 0)$	$(0, 0, 1)$	$(0, 1, 1)$
			C_1	$(1, 1, 1)$	$(0, 1, 0)$	$(0, 0, 1)$	$(0, 1, 1)$
			D_1	$(1, 0, 0)$	$(1, 1, 0)$	$(1, 0, 1)$	$(1, 1, 1)$

So, for n prisoners, we have an n -dimension matrix, therefore 2^n n -tuples must be specified. Now, this game can be expressed much more compactly by the following Boolean game $G = (A, V, \pi, \Phi)$: $A = \{1, 2, \dots, n\}$, $V = \{C_1, \dots, C_n\}$ (with $\neg C_i = D_i$ for every i), $\forall i \in \{1, \dots, n\}$, $\pi_i = \{C_i\}$, and $\forall i \in \{1, \dots, n\}$, $\phi_i = \{(C_1 \wedge C_2 \wedge \dots \wedge C_n) \vee \neg C_i\}$.

Here, each player i has two possible strategies: $s_i = \{C_i\}$ and $s'_i = \{\neg C_i\}$. There are 8 strategy profiles for G , including $S_1 = (C_1, C_2, C_3)$ and $S_2 = (\neg C_1, C_2, C_3)$. Under S_1 , players 1, 2 and 3 have their goal satisfied, while S_2 satisfies only the goal of player 1.

Note that this choice of binary utilities implies a loss of generality, but it is essentially a starting point for the study of Boolean games, which moreover will give us lower complexity bounds. See Section 7.

Definition 4 Let $G = (A, V, \pi, \Phi)$ be a Boolean game, with $\Phi = \{\phi_1, \dots, \phi_n\}$ and $A = \{1, \dots, n\}$. Strategy s_i is a winning strategy for i if $\forall s_{-i} \in 2^{\pi_{-i}}, (s_{-i}, s_i) \models \phi_i$.

Proposition 1 Let $G = (A, V, \pi, \Phi)$ be a Boolean game. Player $i \in A$ has a winning strategy iff $PI_{\pi_i}(\phi_i) \neq \emptyset$.

Clearly enough, deciding the existence of a winning strategy for a given player is an instance of the controllability problem [3, 11] and can be reduced to the resolution of a $\text{QBF}_{2,3}$ instance.

It is also easily seen that Boolean games as studied by Harrenstein et al [8, 7] are a special case of our n -players Boolean games, obtained by making the following two assumptions: $n = 2$ (two players) and $\phi_2 = \neg\phi_1$ (zero-sum).

4 Nash equilibria

Pure-strategy Nash equilibria (PNE) for n -players Boolean games are defined exactly as usual in game theory (see for instance [14]), having in mind that utility functions are induced from players' goals ϕ_1, \dots, ϕ_n . A PNE is a strategy profile such that each player's strategy is an optimum response to the other players' strategies.

Definition 5 Let $G = (A, V, \pi, \Phi)$ be a Boolean game with $A = \{1, \dots, n\}$. $S = \{s_1, \dots, s_n\}$ is a pure-strategy Nash equilibrium (PNE) if and only if $\forall i \in \{1, \dots, n\}, \forall s'_i \in 2^{\pi_i}, u_i(S) \geq u_i(s_{-i}, s'_i)$.

Example 3 Let $G = \{A, V, \pi, \Phi\}$ be the Boolean game defined by $V = \{a, b, c\}$, $A = \{1, 2, 3\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$, $\varphi_1 = \neg a \vee (a \wedge b \wedge \neg c)$, $\varphi_2 = (a \leftrightarrow (b \leftrightarrow c))$ and $\varphi_3 = ((a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c))$.

Player 1 has a winning strategy, namely setting a to false. It can be checked that the strategy profile $S = \{\bar{a}, \bar{b}, c\}$ is the only PNE of G .

In some examples, several PNE appear: in Ex. 1, the PNE are $\{\bar{a}\bar{b}\bar{c}\}$ and $\{\bar{a}\bar{b}\bar{c}\}$, and in Ex. 2, the PNE are $\{C_1C_2C_3\}$ and $\{\bar{C}_1\bar{C}_2\bar{C}_3\}$.

We now give simple characterizations of pure-strategy Nash equilibria in Boolean games, starting with the following one:

Proposition 2 Let $G = (A, V, \pi, \Phi)$ be a Boolean game and let $S \in 2^V$. S is a pure-strategy Nash equilibrium for G iff for all $i \in A$, either $S \models \varphi_i$ or $s_{-i} \models \neg \varphi_i$ holds.

Proof: S is a PNE for G iff $\forall i \in A, \forall s'_i \in 2^{\pi_i}, u_i(S) \geq u_i(s_{-i}, s'_i)$, i.e., $\forall i \in A, \forall s'_i \in 2^{\pi_i}, u_i(S) = 1$ or $u_i(s_{-i}, s'_i) = 0$, i.e., $\forall i \in A, u_i(S) = 1$ or $\forall s'_i \in 2^{\pi_i}, u_i(s_{-i}, s'_i) = 0$. Finally, $u_i(S) = 1 \Leftrightarrow S \models \varphi_i$, and $\forall s'_i \in 2^{\pi_i}, u_i(s_{-i}, s'_i) = 0 \Leftrightarrow \forall s'_i \in 2^{\pi_i}, (s_{-i}, s'_i) \models \neg \varphi_i$, i.e., $s_{-i} \models \neg \varphi_i$. ■

Since $s_{-i} \models \neg \varphi_i$ means that $\neg \varphi_i$ follows from s_{-i} whatever the instantiation of the variables controlled by player i , the previous characterization of PNE can be simplified again, using the forgetting operator.

Proposition 3 Let $S \in 2^V$. S is a pure-strategy Nash equilibrium for G if and only if $S \models \bigwedge_i (\varphi_i \vee (\neg \exists i : \varphi_i))$.

Proof: We have the following chain of equivalences: $s_{-i} \models \neg \varphi_i \Leftrightarrow s_{-i} \models \neg \exists i : \varphi_i \Leftrightarrow (s_i, s_{-i}) \models \neg \exists i : \varphi_i$ (because variables controlled by player i have disappeared from $\neg \exists i : \varphi_i$) $\Leftrightarrow S \models \neg \exists i : \varphi_i$. Putting everything together, we get: $(\forall i \in A, S \models \varphi_i \text{ or } s_{-i} \models \neg \varphi_i) \Leftrightarrow (\forall i \in A, S \models \varphi_i \text{ or } S \models \neg \exists i : \varphi_i) \Leftrightarrow \forall i \in A, S \models \varphi_i \vee (\neg \exists i : \varphi_i) \Leftrightarrow S \models \bigwedge_i (\varphi_i \vee (\neg \exists i : \varphi_i))$. ■

In the particular case of two-players zero-sum Boolean games, we recover the well-known fact that pure-strategy Nash equilibria coincide with winning strategies for one of the players.

Proposition 4 If G is a two-players zero-sum Boolean game, $S = (s_1, s_2)$ is a pure-strategy Nash equilibrium iff s_1 is a winning strategy for 1 or s_2 is a winning strategy for 2.

Proof: Let $S = (s_1, s_2)$ be a PNE. Assume $u_1(S) = 1$ (the case $u_2(S) = 1$ is symmetric). Since G is zero-sum, we have $u_2(S) = 0$. Now since S is a PNE, $\forall s'_2, u_2(S) \geq u_2(s_1, s'_2)$, which entails $\forall s'_2, u_2(s_1, s'_2) = 0$. It follows $\forall s'_2, (s_1, s'_2) \models \neg \varphi_2$, which entails that $\forall s'_2, (s_1, s'_2) \models \varphi_1$. Thus s_1 is a winning strategy for 1.

Conversely, assume that s_1 is a winning strategy for 1 (the case of 2 is symmetric). Then we have $\forall s_2, u_1(s_1, s_2) = 1$ and $\forall s_2, u_2(s_1, s_2) = 0$. Let $S = (s_1, s_2)$ where $s_2 \in 2^{\pi_2}$. We have $\forall s'_1, u_1(S) \geq u_1(s'_1, s_2)$ and $\forall s'_1, u_2(S) \geq u_2(s_1, s'_2)$. Thus S is a PNE. ■

This fact enables us to easily determine the complexity of deciding whether there is a pure-strategy Nash equilibrium in a given Boolean game. Recall that $\Sigma_2^P = \text{NP}^{\text{NP}}$ is the class of all the languages that can be recognized in polynomial time by a nondeterministic Turing machine equipped with NP oracles [15].

Proposition 5 Deciding whether there is a pure-strategy Nash equilibrium in a Boolean game is Σ_2^P -complete. Completeness holds even under the restriction to two-players zero-sum games.

Proof: Membership in Σ_2^P is immediate. Hardness is obtained by a reduction from the problem of deciding the validity of a QBF_{2,3}. Given $Q = \exists A, \forall B, \Phi$, we define a two-players zero-sum Boolean game by $\varphi_1 = \Phi \vee (x \leftrightarrow y)$, where x, y are new variables and $\pi_1 = A \cup \{x\}$. Obviously, this game can be built in polynomial time given Q . Clearly, if Q is valid with $M_A \in 2^A$ as a witness, then both (M_A, x) and (M_A, \bar{x}) are winning strategies for 1. Conversely, if Q is not valid, then whatever $M_A \in 2^A$ 1 plays, 2 can play $M_B \in 2^B$ such that $(M_A, M_B) \not\models \Phi$, and 2 can play \bar{y} (resp. y) if 1 plays x (resp. \bar{x}), resulting in both cases in 2 winning (so, 1 has no winning strategy). Now it is easily seen that 2 has no winning strategy. Finally, there is a winning strategy for 1 (or 2, vacuously) if and only if Q is valid, and Proposition 4 concludes. ■

The fact that this problem lies at the second level of the polynomial hierarchy can intuitively be explained by the presence of two independent sources of complexity: the search for the “good” strategy profiles, and the test whether this strategy profile is indeed a pure-strategy Nash equilibrium. Once again, this result is related to the complexity of controllability [11]. Actually, since the existence of a Nash equilibrium is a more general problem than controllability, the fact that it has the same complexity is rather good news.

We now briefly investigate syntactic restrictions on the formulas representing the players’ goals which make the problem easier. We are especially interested in DNF formulas. Recall that any Boolean function can be represented by such a formula, and thus that this is a syntactic but not a semantic restriction.

As far as 2-players zero-sum games are concerned, since deciding the validity of $\exists A, \forall B, \Phi$, a QBF_{2,3} formula, is Σ_2^P -complete even if Φ is restricted to be in DNF, Proposition 5 holds even if player 1’s goal is restricted to be in DNF (and player 2’s is implicit). However, when we explicitly represent the goals of each player in DNF, the complexity of the problem goes down to NP, as the next proposition shows.

Proposition 6 Let G be a Boolean game. If the goal φ_i of every player is in DNF, then deciding whether there is a pure-strategy Nash equilibrium is NP-complete. Completeness holds even if both we restrict the number of players to 2 and one player controls only one variable.

Proof: If φ_i is in DNF, then $\exists i : \varphi_i$ can be computed in linear time [10, Propositions 17–18]. Thus if every φ_i is in DNF, a formula $\psi \equiv \bigwedge_i (\varphi_i \vee (\neg \exists i : \varphi_i))$ can be computed in linear time. By Proposition 3 it is enough to guess a strategy profile S and check $S \models \psi$, thus the problem is in NP.

As for hardness, we give a reduction from (the complement of) the problem of deciding whether a DNF $\Phi = \bigvee_{i=1}^k T_i$ is tautological, a well-known coNP-complete problem. Write X for the set of variables of Φ and let $x, y \notin X$. Define a two-players game G by $\varphi_1 = \bigvee_{i=1}^k (T_i \wedge x \wedge \neg y) \vee (T_i \wedge \neg x \wedge y)$, $\pi_1 = \{y\}$, $\varphi_2 = (x \wedge y) \vee (\neg x \wedge \neg y)$, $\pi_2 = X \cup \{x\}$. Clearly, G can be built in linear time and φ_1, φ_2 are in DNF. Observe $\varphi_1 \equiv \Phi \wedge (x \neq y)$ and $\varphi_2 \equiv (x = y)$. By Proposition 3, there is a PNE in G if and only if $((\Phi \wedge (x \neq y)) \vee \neg \Phi) \wedge (x = y)$ is satisfiable. Indeed: (i) since y does not occur in Φ we have $\neg \exists y : (\Phi \wedge x \neq y) \equiv \neg (\Phi \wedge \exists y : x \neq y) \equiv \neg (\Phi \wedge \top) \equiv \neg \Phi$, and (ii) $\neg \exists X \cup \{x\} : (x = y) \equiv \perp$. Since $\Phi \wedge (x \neq y) \wedge (x = y)$ is unsatisfiable, there is a PNE in G iff $\neg \Phi \wedge (x = y)$ is satisfiable, i.e., iff $\neg \Phi$ is satisfiable since x and y do

not occur in Φ . Finally, there is a PNE in G iff Φ is not tautological. ■

When restricting to two-players games, the complexity of deciding whether a game has a PNE can even be lowered to P . This is the case if goals are represented in (i) Horn-renamable DNF, (ii) affine form, (iii) 2CNF or (iv) monotone CNF. This is ensured by tractability of projection in these cases, and the same proof as for abduction [17, Section 6] can be used. However, as far as we know the practical interest of these restrictions in our context has not been studied.

5 Dominated strategies

Another key concept in game theory is *dominance*. A strategy s_i for player i strictly dominates another strategy s'_i if it does strictly better than it against all possible combinations of other players' strategies, and weakly dominates it if it does at least as well against all possible combinations of other players' strategies, and strictly better against at least one. The key idea is that dominated strategies are not useful and can be eliminated (iteratively, until a fixpoint is reached). This process relies on the hypothesis that every player behaves in a rational way and knows that the other players are rational.

Definition 6 Let $s_i \in 2^{\pi_i}$ be a strategy for player i . s_i is strictly dominated if $\exists s'_i \in 2^{\pi_i}$ s.t. $\forall s_{-i} \in 2^{\pi_{-i}}$, $u_i(s_i, s_{-i}) < u_i(s'_i, s_{-i})$.

s_i is weakly dominated if $\exists s'_i \in 2^{\pi_i}$ s.t. $\forall s_{-i} \in 2^{\pi_{-i}}$, $u_i(s_i, s_{-i}) \leq u_i(s'_i, s_{-i})$ and $\exists s_{-i} \in 2^{\pi_{-i}}$ s.t. $u_i(s_i, s_{-i}) < u_i(s'_i, s_{-i})$.

The following simple example shows the interest of eliminating dominated strategies.

Example 4 Let $G = \{A, V, \pi, \Phi\}$ be the Boolean game defined by $V = \{a, b\}$, $A = \{1, 2\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\varphi_1 = \varphi_2 = a \wedge \neg b$. This game has two PNE: strategy profiles $S_1 = \{a, \bar{b}\}$ and $S_2 = \{\bar{a}, b\}$. Nevertheless, only one of these equilibria is interesting. Indeed, if 1 and 2 are rational, they will both choose strategy profile S_1 , which makes both of them win. This result may be obtained by eliminating dominated strategies: for player 1 (resp. 2), strategy $\{a\}$ (resp. $\{\bar{b}\}$) weakly dominates strategy $\{\bar{a}\}$ (resp. $\{b\}$). ■

This interest also appears in Ex. 2 (the resulting strategy profile is $\{\overline{C_1 C_2 C_3}\}$), but not in Ex. 1 (the resulting strategy profiles are exactly the PNE). It is a well-known fact from game theory that a strictly dominated strategy is not present in any Nash equilibrium, whereas a weakly dominated strategy can be present in one (see for instance [9].) Moreover, the order of elimination of strictly dominated strategies does not affect the final result, which is no longer true for weakly dominated strategies. Since the latter negative result holds for general games (with no restriction on the players' utility functions), it is worth wondering whether it still holds for Boolean games. It actually does, as shown on the following example.

Example 5 $G = \{A, V, \pi, \Phi\}$, where $V = \{a, b\}$, $A = \{1, 2\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\varphi_1 = a \wedge b$, $\varphi_2 = a \wedge \neg b$. For player 1 (resp. 2), strategy $\{a\}$ (resp. $\{\bar{b}\}$) weakly dominates strategy $\{\bar{a}\}$ (resp. $\{b\}$). If we first eliminate $\{\bar{a}\}$, then $\{\bar{b}\}$ weakly dominates $\{b\}$ and only one strategy profile remains, namely $\{a\bar{b}\}$. Now, if we first eliminate $\{b\}$, then $\{a\}$ no longer dominates $\{\bar{a}\}$ any more, and two strategy profiles remain, namely $\{a\bar{b}\}$ and $\{\bar{a}b\}$.

We now study properties and characterizations of dominating strategies. A first result, that we just state as a remark, is that in a Boolean

game, if strategy s_i strictly dominates strategy s'_i , then s_i is a winning strategy for i . Stated in more formal terms, s_i strictly dominates strategy s'_i if and only if: $s_i \models (\neg \exists -i : \neg \varphi_i)$ and $s'_i \models (\neg \exists -i : \varphi_i)$. This shows that, due to the restriction to binary utilities, the notion of strict dominance degenerates and loses its interest. This is however not the case for weak dominance. We have the following simple characterization of weak dominance:

Proposition 7 Strategy s_i weakly dominates strategy s'_i if and only if $(\varphi_i)_{s'_i} \models (\varphi_i)_{s_i}$ and $(\varphi_i)_{s_i} \not\models (\varphi_i)_{s'_i}$.

Proof: Strategy s_i weakly dominates s'_i iff (i) $\forall s_{-i} \in 2^{\pi_{-i}}, u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$ and (ii) $\exists s_{-i} \in 2^{\pi_{-i}}, u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$.

Now (i) $\Leftrightarrow \forall s_{-i} \in 2^{\pi_{-i}}, (u_i(s_i, s_{-i}) = 1 \text{ or } u_i(s'_i, s_{-i}) = 0) \Leftrightarrow \forall s_{-i} \in 2^{\pi_{-i}}$, if $(s'_i, s_{-i}) \models \varphi_i$ then $(s_i, s_{-i}) \models \varphi_i \Leftrightarrow \forall s_{-i} \in 2^{\pi_{-i}}$, if $s_{-i} \models (\varphi_i)_{s'_i}$ then $s_i \models (\varphi_i)_{s_i} \Leftrightarrow (\varphi_i)_{s'_i} \models (\varphi_i)_{s_i}$.

Finally, (ii) $\Leftrightarrow \neg(\text{i})$ if we swap s_i and s'_i ; thus (ii) $\Leftrightarrow (\varphi_i)_{s_i} \not\models (\varphi_i)_{s'_i}$. ■

Like for Nash equilibria, this characterization allows us to derive the complexity of deciding weak dominance.

Proposition 8 Deciding whether a given strategy s'_i is weakly dominated is Σ_2^P -complete. Hardness holds even if φ_i is restricted to be in DNF.

Proof: Membership in Σ_2^P is immediate. Hardness is obtained again by a reduction from the problem of deciding the validity of a QBF_{2,3}. Given $Q = \exists A, \forall B, \Phi$, let a, b be two new variables, and define $\varphi_1 = (a \wedge \Phi) \vee (\neg a \wedge b)$, $\pi_1 = A \cup \{a\}$, $\pi_2 = B \cup \{b\}$ (φ_2 does not matter). Let M'_A be any A -interpretation and s'_1 be (M'_A, \overline{a}) . We have $(\varphi_1)_{s'_1} \equiv (b)$.

Assume Q is valid with $M_A \in 2^A$ as a witness, and let $s_1 = (M_A, a)$. Then clearly s_1 is a winning strategy for 1 whereas s'_1 is not, thus s_1 weakly dominates s'_1 . Conversely, assume Q is not valid, and let $M_A \in 2^A$. Let $s_1 = (M_A, \overline{a})$. Then $(\varphi_1)_{s_1} \equiv (b) \equiv (\varphi_1)_{s'_1}$, thus by Proposition 7, s_1 does not weakly dominate s'_1 . Now let $s_1 = (M_A, a)$. Since Q is not valid, there is $M_B \in 2^B$ such that $(M_A, M_B) \not\models \Phi$. Thus $(M_B, b) \models (\varphi_1)_{s'_1}$ but $(M_B, b) \not\models (\varphi_1)_{s_1}$, and by Proposition 7, s_1 does not weakly dominate s'_1 . Finally, s'_1 is weakly dominated iff Q is valid. For goals in DNF, just note (i) if Φ is in DNF then $\exists A, \forall B, \Phi$ is still Σ_2^P -complete and (ii) a DNF for φ_1 can be built efficiently. ■

6 Related work

Our work is not the first one that gives a logical account to the study of concepts such as Nash equilibria and dominating strategies. Apart from Boolean games [8, 7, 5], a number of other works considered static games from the point of view of logic and AI.

Two recent lines of work allow for expressing games with *ordinal* preferences within well-developed AI frameworks.

In Foo et al [6], a game in normal form is mapped into a *logic program with ordered disjunction* (LPOD) where each player owns a set of clauses that encode the player's preference over her possible actions given every possible strategy profile of other players. It is then shown that pure-strategy Nash equilibria correspond exactly to the most preferred answer sets. The given translation suffers from a limitation, namely its size: the size of the LPOD is the same as that of the normal form of the game (since each player needs a number of clauses equal to the number of possible other strategy profiles for other players). However, this limitation is due to the way LPODs are

induced from games and could be overwhelmed by allowing to express the players' preferences by any LPODs (and prioritized goals), which then would allow for a possible much more compact representation.

In De Vos et al [4], a strategic game is represented using a *choice logic program*, where a set of rules express that a player will select a "best response" given the other players' choices. Then, for every strategic game, there exists a choice logic program such that the set of stable models of the program coincides with the set of Nash equilibria of the game. This property provides a systematic method to compute Nash equilibria for finite strategic games.

In Apt et al [1], CP-nets are viewed as games in normal form and vice versa. Each player i corresponds to a variable X_i of the CP-net, whose domain $D(X_i)$ is the set of available actions to the player. Preferences over a player's actions given the other players' strategies are then expressed in a conditional preference table. The CP-net expression of the game can sometimes be more compact than its normal form explicit representation, provided that some players' preferences depend only on the actions of a subset of other players. A first important difference with our framework is that we allow players to control an arbitrary set of variables, and thus we do not view players as variables; the only way of expressing in a CP-net that a player controls several variables would consist in introducing a new variable whose domain would be the set of all combination of values for these variables—and the size of the CP-net would then be exponential in the number of variables. A second important difference, which holds as well for the comparison with Foo et al [6] and De Vos et al [4], is that players can express arbitrary binary preferences, including extreme cases where the satisfaction of a player's goal may depend only of variables controlled by other players. A last (less technical and more foundational) difference with both lines of work, which actually explains the first two above, is that we do not map normal form games into anything but we *express* games using a logical language.

Admittedly, on the other hand, the restriction to binary preferences is a strong limitation, but our work can be viewed as a preliminary, but necessary step, and the extension of our notions, and of some of our results, to non-degenerated (ordinal or numerical) preferences do not present any particular problem (see Section 7).

In Section 4 we mentioned a relationship to propositional controllability, as studied by Boutilier [3] and Lang et al [11]. A recent line of work within this alley [16] studies a cooperation logic in which each agent is assumed to control a set of propositional variables. While we focus on preferences and solution concepts, van der Hoek and Wooldridge [16] focus on the effective power of agents, that is, they reason about the state of affairs that a group of agents can bring about.

7 Conclusion

In this paper we extended Boolean games to an arbitrary number of players and to arbitrary Boolean goals. Extended Boolean games are a first step towards a more general framework for expressing and reasoning with interacting agents when the set of strategy profiles has a combinatorial structure.

Clearly, the restriction to a single goal for each agent—and therefore to binary utilities—is a strong one. However, as often, proceeding by "building blocks" is valuable, for at least two reasons. First, once this Boolean games framework is defined, extending it so as to allow for more general preferences does not present any particular difficulty: the definition remains unchanged *except* the agents' goals $\varphi_1, \dots, \varphi_n$, which are replaced by more complex structures, expressed within

logical languages for compact representation. These frameworks allow for representing compactly either *numerical* preferences (utility functions on 2^V) or *ordinal* preferences (partial or complete orderings on 2^V). The companion paper [2] considers extended Boolean games with ordinal preferences represented by prioritized goals and CP-nets with binary variables.

Now, binary utilities are a degenerate case of both numerical and ordinal preferences. This implies that the complexity results identified in this paper provide *lower bounds* for complexity results in the aforementioned possible extensions of the framework⁵.

Apart of extensions to more expressive preferences as explained above, we plan to address the following two issues:

- computing *mixed strategy Nash equilibria* for Boolean games;
- defining and studying *dynamic Boolean games* (with complete or incomplete information).

REFERENCES

- [1] K. R. Apt, F. Rossi, and K. B. Venable. CP-nets and Nash equilibria. In Elsevier, editor, *Proc. CIRAS 2005 (Third International Conference on Computational Intelligence, Robotics and Autonomous Systems)*, Singapore, December 13–16 2005.
- [2] E. Bonzon, M.-C. Lagasquie-Schiex, and J. Lang. Compact preference representation for Boolean games. In *Proceedings of PRICAI-06*, 2006. To appear.
- [3] C. Boutilier. Toward a logic for qualitative decision theory. In *KR-94*, pages 75–86, 1994.
- [4] M. De Vos and D. Vermeir. Choice logic programs and Nash equilibria in strategic games. In Jorg Flum and Mario Rodriguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, volume 1683, pages 266–276, 1999.
- [5] P.E. Dunne and W. van der Hoek. Representation and complexity in boolean games. In *Proc. of JELIA2004*, volume LNCS 3229, pages 347–359. José Júlio Alferes et João Alexandre Leite (eds), 2004.
- [6] N. Foo, T. Meyer, and G. Brewka. LPOD answer sets and Nash equilibria. In M. Maher, editor, *Proceedings of the 9th Asian Computer Science Conference (ASIAN 2004)*, pages 343–351. Chiang Mai, Thailand, Springer LNCS 3321, 2004.
- [7] P. Harrenstein. *Logic in Conflict*. PhD thesis, Utrecht University, 2004.
- [8] P. Harrenstein, W. van der Hoek, J.J. Meyer, and C. Witteveen. Boolean games. In J. van Benthem, editor, *Proceedings of TARK 2001*, pages 287–298. Morgan Kaufmann, 2001.
- [9] J. Hillas and E. Kohlberg. Foundations of strategic equilibrium. In R. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 3, pages 1598–1663. North-Holland, 2002.
- [10] J. Lang, P. Liberatore, and P. Marquis. Propositional independence - formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.
- [11] J. Lang and P. Marquis. Two forms of dependence in propositional logic: controllability and definability. In *Proceedings of AAAI-98*, pages 268–273, 1998.
- [12] F. Lin. On the strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128:143–159, 2001.
- [13] F. Lin and R. Reiter. Forget it. In *In proc. of the AAAI falls symposium on Relevance*, pages 154–159, 1994.
- [14] M. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.
- [15] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [16] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, 2005.
- [17] B. Zanuttini. New polynomial classes for logic-based abduction. *Journal of Artificial Intelligence Research*, 19:1–10, 2003.

⁵ For Proposition 5, this lower bound will also be an upper bound for most extensions of the framework.

An Automated Agent for Bilateral Negotiation with Bounded Rational Agents with Incomplete Information¹

Raz Lin² and Sarit Kraus^{2,3} and Jonathan Wilkenfeld^{3,4} and James Barry⁵

Abstract. Many day-to-day tasks require negotiation, mostly under conditions of incomplete information. In particular, the opponent's exact tradeoff between different offers is usually unknown. We propose a model of an automated negotiation agent capable of negotiating with a bounded rational agent (and in particular, against humans) under conditions of incomplete information. Although we test our agent in one specific domain, the agent's architecture is generic; thus it can be adapted to any domain as long as the negotiators' preferences can be expressed in additive utilities. Our results indicate that the agent played significantly better, including reaching a higher proportion of agreements, than human counterparts when playing one of the sides, while when playing the other side there was no significant difference between the results of the agent and the human players.

1 Introduction

Many day-to-day tasks require negotiation, often in a bilateral context. In this setting two agents are negotiating over one or several issues in order to reach consensus [4, 6, 7]. The sides might have conflicting interests, expressed in their utility functions, and they might also cooperate in order to reach beneficial agreements for both sides [1, 12]. While models for bilateral negotiations have been extensively explored, the setting of bilateral negotiation between an automated agent and a bounded rational agent is an open problem for which an adequate solution has not yet been found. Despite this, the advantages of succeeding in presenting an automated agent with such capabilities cannot be understated. Using such an agent could help in training people in negotiations and assist in e-commerce environments by representing humans and other activities in daily life. Our proposed automated agent makes a significant contribution in this respect. An even more difficult problem occurs when incomplete information is added into this environment. That is, there is also lack of information regarding some parameters of the negotiation.

We consider a setting of a finite horizon bilateral negotiation with incomplete information between an automated agent and a bounded rational agent. The incomplete information is expressed as uncertainty regarding the utility preferences of the opponent, and we assume that there is a finite set of different agent types. The negotiation itself consists of a finite set of multi-attribute issues and time-constraints. If no agreement is reached by the given deadline a status-

quo outcome is enforced. We propose a model of an automated agent for this type of negotiation by using a qualitative decision making approach [2, 13]. In our experiments we matched our automated agent against human negotiators. By analyzing the results of the experiments, we show that our agent is capable of negotiating efficiently and reaching multi-attribute agreements in such an environment. When playing one of the sides in the negotiation our agent reached significantly better results than the human players, and also allowed both sides to reach an agreement significantly faster. On the other hand, while our agent was playing the other side, though it did not reach significantly better results than the human player, it did not reach worse results. Also, there are no significant differences between the results reached by the human players and the automated agent playing that role.

The rest of the paper is organized as follows. Section 2 provides an overview of bilateral negotiation with incomplete information. Section 3 surveys related work done in the field of negotiation with incomplete information and bounded rational agents. Section 4 presents the design of our automated agent, including its beliefs updating and decision making mechanisms. Section 5 describes the experimental setting and methodology and reviews the results. Finally, Section 6 provides a summary and discusses future work.

2 Problem Description

We consider a bilateral negotiation in which two agents negotiate to reach an agreement on conflicting issues. The negotiation can end either when (a) the negotiators reach a full agreement, (b) one of the agents opts out, thus forcing the termination of the negotiation with an opt-out outcome denoted OPT , or (c) a predefined deadline is reached, denoted dl , in which, if a partial agreement was reached it is implemented or, if no agreement was reached, a status quo outcome, denoted SQ , is implemented. Let I denote the set of issues in the negotiation, o_i the finite set of values for each $i \in I$ and $O = o_1 \times o_2 \times \dots \times o_{|I|}$ a finite set of values for all issues. Since we allow partial agreements $\emptyset \in o_i$ for each $i \in I$. An offer is then denoted as a vector $\vec{o} \in O$. It is assumed that the agents can take actions during the negotiation process until it terminates. Let **Time** denote the set of time periods in the negotiation, that is **Time** = {0, 1, ..., dl }. Time also plays a factor and has an influence on the agents' utilities. Each agent is assigned with a time cost which influences his utility as time passes.

In each period $t \in \text{Time}$ of the negotiation, if the negotiation has not terminated earlier, each agent can propose a possible agreement, and the opponent can either accept or reject the offer. Each agent can either propose an agreement which consists of all the issues in the negotiation, or a partial agreement. In contrast to the model of

¹ This research was supported in part by NSF under grant #IIS-0208608.

² Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel 52900. email: {linraz,sarit}@cs.biu.ac.il

³ Institute of Advanced Computer Studies, University of Maryland.

⁴ Department of Government and Politics, University of Maryland, College Park, Maryland. email: jwilkenf@gvpt.umd.edu

⁵ Center for International Development and Conflict Management, University of Maryland, College Park, Maryland. email: jbarry6899@aol.com

alternating offers [12], each agent can perform up to M interactions with the opponent agent in a given time period. Thus, an agent must worry that its opponent may opt out in any time period.

Since we deal with incomplete information, we assume that there is a finite set of agent types. These types are associated with different additive utility functions. Formally, we denote the possible types of the agents $\text{Types} = \{1, \dots, k\}$. We refer to an agent whose utility function is u_l as an agent of type l , and $u_l : \{(O \cup \{SQ\} \cup \{OPT\}) \times \text{Time}\} \rightarrow \mathbb{R}$. Each agent is given its exact utility function. The agent, and the subjects in the experiments described later in the paper, are also aware of the set of possible utility functions of the opponent. However, the exact utility function of the rival is private information. Our agent has some probabilistic belief about the type of the other agent. This belief may be updated over time during the negotiation process (for example, using Bayes formula).

3 Related Work

The problem of modeling an automated agent for bilateral negotiation is not new for researchers in the fields of Multi-Agent Systems and Game Theory. However, most research makes simplifying assumptions that do not necessarily apply in genuine negotiations, such as assuming complete information [3, 11] or the rationality of the opponent [3, 4, 5, 7]. None of the above researchers has looked into the negotiation process in which there is both incomplete information and the opponent is bounded rational. While their approaches might be appropriate in their context, they cannot be applied to our settings.

To deal with the opponent's bounded rationality, researchers suggested shifting from quantitative decision theory to qualitative decision theory [13]. In using such a model we do not necessarily assume that the opponent will follow the equilibrium strategy or try to be a utility maximizer. Also, this model is better suited for cases in which the utility or preferences are unknown but can be expressed in ordinal scales or as preference relations [2]. This approach seems appropriate in our settings, and using the maximin criteria, which is generally used in this context, enables our agent to follow a pessimistic approach regarding the probability that an offer will be accepted.

Another way to deal with bounded rationality was suggested by Luce [8], who introduced the notion of *Luce numbers*. A Luce number is a non-negative number that is attached to each offer. The Luce number of an offer $\vec{o}^* \in O$ is calculated using the following formula:

$$lu(o^*) = \frac{u(\vec{o}^*)}{\sum_{\vec{o} \in O} u(\vec{o})} \quad (1)$$

That is, the Luce numbers assign probabilistic beliefs regarding the acceptance of the offer by the opponent.

Several methods are proposed when dealing with the incomplete information regarding the preferences of an opponent. For example, Bayes theorem is the core component of the *Bayesian Nash equilibrium* ([12], p. 24-29), and it is used to deduce the current state given a certain signal. It allows us to compensate for incomplete information and enables good adaptation in a negotiation with time-constraints. In finite horizon negotiations there are no past interactions to learn from and not enough time periods to build a complete model. Thus this model supplies a good probabilistic tool to model the opponent, as opposed to neural networks [11] or genetic algorithms [7]. In our settings, Bayes theorem can be used to update the believed type of the opponent and at each time period act as if the opponent is of a certain type.

4 Agent Design

Our agent is built with two mechanisms: (a) a decision making mechanism, and (b) a mechanism for updating beliefs. We describe both mechanisms in the following subsections.

4.1 The Qualitative Decision Making Component (QDM)

The qualitative decision making component takes into account the agent's utility function, as well as the believed type of the opponent. This data is used both for deciding whether to accept or reject an offer and for generating an offer. In our settings, although several offers can be proposed in each time period, we restrict our agent to making a single offer in each period. This is done due to the fact that our mechanism for generating offers only produces one distinct offer at a given time period. The bounded rational agent, on the other hand, is free to propose several offers, and our agent can respond to all the offers, which indeed happened in the simulations. While we provide some theoretical foundations for our approach, we demonstrate its effectiveness using simulations with human subjects in an environment of incomplete information, as described in Section 5.

4.1.1 Generating Offers

The motivation behind the mechanism for generating offers is that the automated agent would like to propose an offer, which yields him the highest utility value. However, due to conflict of interests, there is a high probability that this agreement will be rejected by the opponent. To overcome this, our agent uses a qualitative decision strategy. Basically, the agent evaluates all possible offers based on its utility and the probability that the rival will accept them. *Luce numbers* are used to estimate this probability. We note that for every two offers x and y and agent j , where $lu_j(x)$ and $lu_j(y)$ denote the Luce numbers associated with offers x and y respectively for agent j , if $u_j(x) \geq u_j(y)$ then $lu_j(x) \geq lu_j(y)$. That is, the higher the Luce number of an offer, the greater the probability of it being accepted.

Since the opponent himself also tries to reason whether an offer will be accepted by our agent, we take the Luce numbers of both our agent and the opponent into account. That is, our agent tries to estimate, from the opponent's point of view, whether the opponent will accept the offer. This is done by calculating the sum of the Luce numbers of the agent and the opponent. This sum is used as an estimation for the acceptance of the offer, and is multiplied by the utility value of the opponent from that offer. Finally, our agent compares those values with its own utility values. Similarly to the qualitative decision theory, which uses the maximin value [2, 13], our agent selects the minimum value between those two values, under the pessimistic assumption that the probability that an offer is accepted is based on the agent that favors the offer the least. After calculating the minimum value between all the offers, our agent selects the offer with the maximum value among all the minima, in order to also try and maximize its own utility. Thus, our qualitative offer generation mechanism selects, intuitively, the best offer among the offers that the agent believes that might be accepted. In the rest of this section we describe this process formally.

We assume that, at each time period t , the automated agent has a belief about the type of its opponent. This believed type, denoted by $BT(t)$, is the one that the agent presumes to be the most probable for the opponent. The agent uses the utility function associated with that type in all of the calculations in that time period. In Section 4.2 we

describe in detail the elicitation of this belief. We denote by $u_{opp}^{BT(t)}$ the utility function associated with the believed type of the opponent at time t . From this utility function, our agent derives the *Luce numbers* [8]. Since the Luce number is calculated based on a given utility function, we denote the Luce number of an offer derived from the opponent's believed utility at time t , $BT(t)$, by $lu_{opp}(\text{offer} \mid BT(t))$, and the Luce number for an offer derived by the agent's own utility simply as $lu(\text{offer})$. We denote our function by $QO(t)$ (standing for Qualitative Offer), where $t \in \text{Time}$ is the time of the offer. If the current agent is j , the strategy selects an offer o in time t such that:

$$QO(t) = \arg \max_{o \in O} \min\{\alpha, \beta\} \quad (2)$$

$$\text{where } \alpha = u_j(o, t)$$

$$\text{and } \beta = [lu_{opp}(o \mid BT(t)) + lu(o)] \cdot u_{opp}^{BT(t)}(o, t)$$

Seemingly, our QO function is a non-classical method for generating offers. However, not only were we able to show its efficacy by empirical experiments, in which it was used in negotiations with bounded rational agents, as we describe in Section 5, we also showed (Section 4.1.3) that it also conforms to some properties from classical negotiation theory, which are mainly used by mediators. Note that the formula does not build on the bounded rationality of the opponent.

The next subsection deals with the question of when the agent should accept an incoming offer or reject it.

4.1.2 Accepting Offers

The agent needs to decide what to do when it receives an offer from its opponent, offer_{opp} , at time $t - 1$. If we refer to the automated agent as agent 1 and the bounded rational agent as agent 2, if $u_1(\text{offer}_{opp}) \geq u_1(QO(t))$ then our agent accepts the offer. Otherwise, our agent should not immediately rule out accepting the offer it has just received. Instead, it should take into consideration the probability that its counter-offer will be accepted or rejected by the opponent. This is done by comparing the believed utility of the opponent from the original offer as compared with the opponent's utility from our offer. If the difference is lower than a given threshold⁶ T , that is $|u_2(QO(t)) - u_2(\text{offer}_{opp})| \leq T$, then there is a high probability that the opponent will be indifferent between its original offer and our counter-offer, so our agent will reject the offer and propose a counter-offer (taking a risk that the offer will be rejected), since the counter-offer has a better utility value for our agent. If the difference is greater than the threshold, i.e., there is a higher probability that the opponent will not accept our counter-offer, our agent will accept the opponent's offer with a given probability, which is attached to each outcome. To this end we define the rank number, which is associated with each offer and a given utility function u , denoted *rank*(offer). The rank number of an offer is calculated by ordering all offers on an ordinal scale between 1 and $|O|$ according to their utility values, and dividing the offer's ordering number by $|O|$. That is, the agent will accept an offer with a probability $rank(\text{offer}_{opp})$ and reject and make a counter-offer $QO(t)$ with probability $1 - rank(\text{offer}_{opp})$. The intuition behind this is to enable the agent also to accept agreements based on their relative values, on an ordinal scale of $[0..1]$, and not based on their absolute values.

In the next subsection we demonstrate that our proposed solution also conforms to some properties of the *Nash bargaining solution*. This gives us the theoretical basis for the usage of our technique in

bilateral negotiation, and for the assumption that offers proposed by our agent will also be considered to be accepted by the opponent.

4.1.3 QO: An Alternative to Nash Bargaining Solution

We employ from Luce and Raiffa [9] the definitions of a bargaining problem and the Nash bargaining solution. We denote by $B = (u_1(\cdot), u_2(\cdot))$ the bargaining problem with two utilities, u_1 and u_2 . The Nash bargaining solution (note that the solution is not the offer itself, but rather the payoff of the offer) is defined by several characteristics and is usually designed for a mediator in an environment with complete information. A bargaining (or a negotiation) solution f should satisfy symmetry, efficiency, invariance and independence of irrelevant alternatives, wherein *symmetry* states that if both players have the same bargaining power, then neither player will have any reason to accept an agreement which yields a lower payoff for it than for its opponent. For example, for the solution to be symmetric, it should not depend on the agent who started the negotiation process. *Efficiency* states that two rational agents will not agree on an agreement if its utility is lower for both of them than another possible agreement. This solution is said to be Pareto-optimal. *Invariance* states that for all equivalent problems B and B' , that is $B' = (\alpha_1 + \beta_1 \cdot u_1(\cdot), \alpha_2 + \beta_2 \cdot u_2(\cdot))$, $\alpha_1, \alpha_2 \in \mathbb{R}$, $\beta_1, \beta_2 \in \mathbb{R}^+$, the solution is also the same, $f(B) = f(B')$. That is, two positive affine transformations can be applied on the utility functions of both agents and the solution will remain the same. Finally, *independence of irrelevant alternatives* asserts that the solution $f(B) = f(B')$ whenever $B' \subseteq B$ and $f(B) \subseteq B'$. That is, if new agreements are added to the problem in such a manner that the status quo remains unchanged, either the original solution is unchanged or it becomes one of the new agreements.

It was shown by Nash [10] that the only solution that satisfies all of these properties is the product maximizing of the agents' utilities. However, as we stated, the Nash solution is usually designed for a mediator. Since we propose a model for an automated agent which negotiates with bounded rational agents following the QO function (Equation 2), our solution cannot satisfy all of these properties. To this end, we modified the independence of irrelevant alternatives property to allow for a set of possible solutions instead of one unique solution:

- **PROPERTY 4A (Independence of irrelevant alternatives solutions)**
A negotiation solution f satisfies independence of irrelevant alternatives solutions if the set of all possible solutions of $f(B)$ is equal to the set of all possible solutions of $f(B')$ whenever $B' \subseteq B$ and $f(B) \subseteq B'$.

Proving that our agent's strategy for proposing offers conforms to those properties is important since although the agent should maximize its own utility, it should also find agreements that would be acceptable to its opponent.

Theorem 1 *The QO function satisfies the properties of symmetry, efficiency and independence of irrelevant alternatives solutions.*

Due to space limitations, we do not present the proof of the theorem. We also showed that in certain cases QO generates agreements which are better for the automated agent than agreements that would have been generated by following the Nash solution.

4.2 The Bayesian Updating Rule Component

The Bayesian updating rule (BUR) is based on Bayes Theorem described above and it provides a practical learning algorithm. We as-

⁶ In the simulations, T was set to 0.05.

sert that there are several types (or clusters) of possible agents. The bounded rational agent should be matched to one such type. In each time period, our agent consults the BUR component in order to update its belief regarding the opponent's type.

Recall that there are k possible types for an agent. At time $t = 0$ the prior probability of each type is equal, that is, $P(\text{type}) = \frac{1}{k}, \forall \text{type} \in \text{Types}$. Then, for each time period t we calculate the posteriori probability for each of the possible types, taking into account the history of the negotiation. This is done incrementally after each offer is received or accepted. Then, this value is assigned to $P(\text{type})$. Using the calculated probabilities, the agent selects the type whose probability is the highest and proposes an offer as if this is the opponent's type. Formally, at each time period $t \in \text{Time}$ and for each type $\in \text{Types}$ and offert _{t} (the offer at time period t) we compute:

$$P(\text{type}|offert_t) = \frac{P(\text{offert}_t|\text{type})P(\text{type})}{P(\text{offert}_t)} \quad (3)$$

where $P(\text{offert}_t) = \sum_{i=1}^{|\text{Types}|} P(\text{offert}_t|\text{type}_i) \cdot P(\text{type}_i)$. Since the Luce numbers actually assign probabilities to each offer, $P(\text{offert}_t|\text{type})$ is computed using the Luce numbers.

Now we can deduce the believed type of the opponent for each time period t , denoted as $BT(t)$, using the following equation:

$$BT(t) = \arg \max_{\text{type} \in \text{Types}} P(\text{type}|offert_t), \quad \forall t \in \text{Time} \quad (4)$$

Using this updating mechanism enables our updating component to conform to the following conditions, which are generally imposed on an agent's system of beliefs, and which are part of the conditions for a sequential Bayesian equilibrium [12]: (a) *consistency* and (b) *never dissuaded once convinced*. *Consistency* implies that agent j 's belief should be consistent with its initial belief and with the possible strategies of its opponents. These beliefs are updated whenever possible, while *never dissuaded once convinced* implies that once an agent is convinced of its opponent's type with a probability of 1, or convinced that its opponent cannot be of a specific type, it is never dissuaded from its view. The results of the simulation indeed show that in more than 70% of the simulations our agent believed that its opponent is of the correct type with a probability of 1 or with the highest probability amongst the other possible types.

5 Experiments

We developed a simulation environment which is adaptable such that any scenario and utility functions, expressed as multi-issue attributes, can be used, with no additional changes in the configuration of the interface of the simulations or the automated agent. Our agent can play either role in the negotiation, while the human counterpart accesses the negotiation interface via a web address. The negotiation itself is conducted using a semi-formal language. Each agent constructs an offer by choosing the different values constituting the offers. Then, the offer is constructed and sent in plain English to its counterpart.

To test the efficiency of our proposed agent, we have conducted experiments on a specific negotiation domain⁷. In the following subsections we describe our domain, the experimental methodology and review the results.

⁷ To show that our agent is capable of negotiating in other domains as well, we loaded another domain and tested the agent. As expected, the agent performs as well as in the specified domain. That is, only the utility functions play a role, and not the scenario or the domain.

5.1 Experimental Domain

The experimental domain adheres to the problem definitions described in Section 2. In our scenario England and Zimbabwe are negotiating in order to reach an agreement growing out of the World Health Organization's Framework Convention on Tobacco Control, the world's first public health treaty. The principal goal of the convention is "to protect present and future generations from the devastating health, social, environmental and economic consequences of tobacco consumption and exposure to tobacco smoke." There are three possible agent types, and thus a set of six different utility functions was created. This set describes the different types or approaches towards the negotiation process and the other party. For example, type (a) has a long term orientation regarding the final agreement, type (b) has a short term orientation, and type (c) has a compromise orientation.

Each negotiator was assigned a utility function at the beginning of the negotiation but had incomplete information regarding the opponent's utility. That is, the different possible types of the opponent were public knowledge, but the exact type of the opponent was unknown. The negotiation lasts at most 14 time periods, each with a duration of two minutes. If an agreement is not reached by the deadline then the negotiation terminates with a status quo outcome. Each party can also opt out of the negotiation if it decides that the negotiation is not proceeding in a favorable way. Opting out by England means trade sanctions imposed by England on Zimbabwe (including the ban on the import of tobacco from Zimbabwe), while if Zimbabwe opts out then it will boycott all British imports.

A total of 576 agreements exist, consisting of the following issues: (a) the total amount to be deposited into the Tobacco Fund to aid countries seeking to rid themselves of economic dependence on tobacco production, (b) impact on other aid programs, (c) trade issues, and (d) creation of a forum to explore comparable arrangements for other long-term health issues. While on the first two issues there are contradicting preferences for England and Zimbabwe, for the last two issues there are options which might be preferred by both sides.

5.2 Experimental Methodology

We tested our agent against human subjects, all of whom are computer science undergraduates. The experiment involved 44 simulations with human subjects, divided into 22 pairs. Each simulation was divided into two parts: (i) negotiating against another human subject, and (ii) negotiating against the automated agent. The subjects did not know in advance against whom they played. Also, in order not to bias the results as a consequence of the subjects getting familiar with the domain and the simulation, for exactly half of the subjects the first part of the simulation consisted of negotiating with a human opponent, while the other half negotiated first with the automated agent. The outcome of each negotiation is either the reaching of a full agreement, opting out, or reaching the deadline.

5.3 Experimental Results

The main goal of our proposed method was to enable the automated agent to achieve higher utility values at the end of the negotiation than the human negotiator, playing the same role, achieved, and to allow the negotiation process to end faster as compared to negotiations without our agent. Another secondary goal was to test whether our agent increased the social welfare of the outcome.

Table 1 summarizes the average utility values of all the negotiations and the average of the sums of utility values in all the experiments. HZ and HE denote the utility value gained by the human

playing the role of Zimbabwe or England, respectively, and QZ and QE denote the utility value gained by the *QO* agent playing either role. The utility values ranged from -575 to 895 for the England role and from -680 to 830 for the Zimbabwe role. The Status-Quo value in the beginning of the negotiation was 150 for England and -610 for Zimbabwe. England had a fixed gain of 12 points per time period, while Zimbabwe had a fixed loss of -16 points.

Table 1. Final negotiations utility values and sums of utility values

Parameter	Avg	Stdev
QZ vs. HE	-27.4	287.2
HZ vs. HE	-260	395.6
QE vs. HZ	150.5	270.8
HE vs. HZ	288.0	237.1
HZ vs. QE	113.09	353.23
HE vs. QZ	349.14	299.36
Sum - HE vs. QZ	321.7	223.4
Sum - HZ vs. QE	263.6	270.5
Sum - HE vs. HZ	27.86	449.9

First, we examined the final utility values of all the negotiations for each player, and the sums of the final utility values. The results show that when the automated agent played the role of Zimbabwe, it achieved significantly higher utility values as opposed to a human agent playing the same role (using paired *t-test*: $t(22) = 2.23, p < 0.03$). On the other hand, when playing the role of England, there is no significant difference between the utility values of our agent and the human player, though the average utility value for the human was higher than for the automated agent. The results also show that the average utility values when the human played against another human are lower than the average utility values when he played against our agent. However, these results are significant only when the human players played the role of Zimbabwe ($t(22) = -3.43, p < 0.003$). One explanation for the higher values achieved by the *QO* agent is that the *QO* agent is more eager to accept agreements than humans. When playing Zimbabwe side, which has a negative time cost, accepting agreements sooner, rather than later, allowed the agent to gain higher utility values than the human playing the same side.

Comparing the sum of utility values of both negotiators, based on the role our agent played, we show that this sum is significantly higher when the negotiations involved our agent (either when our agent played the role of Zimbabwe or England), as opposed to negotiations involving only human players (using 2-sample *t-test*: $t(22) = 2.74, p < 0.009$ and $t(22) = 2.11, p < 0.04$).

Another important aspect of the negotiation is the outcome - whether a full agreement was reached or whether the negotiation ended with no agreement (either status-quo or opting out) or with a partial agreement. While only 50% of the negotiations involving only humans ended with a full agreement, 80% of the negotiations involving the automated agent ended with a full agreement. Using *Fishers Exact test* we determined that there is a correlation between the kind of the opponent agent (be it our agent or the human) and the form of the final agreement (full, partial or none). The results show that there is a significantly higher probability of reaching a full agreement when playing against our agent ($p < 0.006$).

Then we examined the final time period of the negotiations. All of the test results showed that in negotiations in which our agent was involved the final time period was significantly lower than the final time period in negotiations in which only humans were involved. The average final time period with two human negotiators was 11.36, while the average final time period against our agent playing the role of Zimbabwe and the role of England was 6.36 and 6.27 respectively.

We used the 2-sample Wilcoxon test to compare between the final time periods of our agent and the human agent when playing against the same opponent (for England role $p < 0.001$ and for Zimbabwe role $p < 0.002$). In addition, we used the Wilcoxon signed rank test, to examine the final time period achieved by the same human player in the two simulations in which he participated - once involving another human player, and another involving an automated agent (for England role $p < 0.0004$, for Zimbabwe role $p < 0.001$).

We also examined the total number of offers made and received in each negotiation. The average number of offers made and received in negotiations involving our agent was 10.86 and 10.59, while in negotiation involving only human players the average number was 18.09. Using 2-sample Wilcoxon test we show that significantly fewer offers were made when our agent was involved ($p < 0.007$ and $p < 0.004$).

6 Conclusions

This paper outlines an automated agent design for bilateral negotiation with bounded rational agents where there is incomplete information regarding the opponent's utility preferences. The automated agent incorporated a qualitative decision making mechanism. The results showed that our agent is indeed capable of negotiating successfully with human counterparts and reaching efficient agreements. In addition, the results demonstrated that the agent played at least as well as, and in the case of one of the two roles, gained significantly higher utility values, than the human player. Though the experiments were conducted on a specific domain, it is quite straightforward to adapt the simulation environment to any other scenario.

Although much time was spent on designing the mechanism for generating an offer, the results showed that most of the agreements reached were offered by the human counterpart. Our agent accepted the offers based on a probabilistic heuristic. A future research direction would be to improve this mechanism and the probabilistic heuristic and then test it by conducting the same sets of experiments.

REFERENCES

- [1] C. Boutilier, R. Das, J. O. Kephart, G. Tesauro, and W. E. Walsh, 'Co-operative negotiation in autonomic systems using incremental utility elicitation', in *Proceedings of UAI-03*, pp. 89–97, (2003).
- [2] D. Dubois, H. Prade, and R. Sabbadin, 'Decision-theoretic foundations of qualitative possibility theory', *European Journal of Operational Research*, **128**, 459–478, (2001).
- [3] P. Faratin, C. Sierra, and N. R. Jennings, 'Using similarity criteria to make issue trade-offs in automated negotiations', *AII*, **142**(2), 205–237, (2002).
- [4] S. Fatima and M. Wooldridge, 'An agenda based framework for multi-issue negotiation', *AII*, **152**, 1–45, (2004).
- [5] S. Fatima, M. Wooldridge, and N. R. Jennings, 'Bargaining with incomplete information', *AMAI*, **44**(3), 207–232, (2005).
- [6] P. Hoz-Weiss, S. Kraus, J. Wilkenfeld, D. R. Andersen, and A. Pate, 'An automated agent for bilateral negotiations with humans', in *Proc. of AAAI/IAAI-02*, pp. 1000–1001, (2002).
- [7] R. J. Lin and S. T. Chou, 'Bilateral multi-issue negotiations in a dynamic environment', in *Workshop on AMEC V*, (2003).
- [8] R. D. Luce, *Individual Choice Behavior: A Theoretical Analysis*, John Wiley & Sons, NY, 1959.
- [9] R. D. Luce and H. Raiffa, *Games and Decisions - introduction and critical survey*, John Wiley & Sons, NY, 1957.
- [10] J. F. Nash, 'The bargaining problem', *Econ.*, **18**, 155–162, (1950).
- [11] M. Oprea, 'An adaptive negotiation model for agent-based electronic commerce', *Studies in Informatics and Control*, **11**(3), 271–279, (2002).
- [12] M. J. Osborne and A. Rubinstein, *A Course In Game Theory*, MIT Press, Cambridge MA, 1994.
- [13] M. Tennenholtz, 'On stable social laws and qualitative equilibrium for risk-averse agents', in *KR*, pp. 553–561, (1996).

Self-Organizing Multiagent Approach to Optimization in Positioning Problems

Sana Moujahed and Olivier Simonin and Abderrafia Koukam¹ and Khaled Ghédira²

Abstract. The facility positioning³ problem concerns the location of facilities such as bus-stops, fire stations, schools, so as to optimize one or several objectives. This paper contributes to research on location problems by proposing a reactive multiagent approach. Particularly, we deal with the p-median problem, where the objective is to minimize the weighted distance between the demand points and the facilities. The proposed model relies on a set of agents (the facilities) situated in a common environment which interact and attempt to reach a global optimization goal: the distance minimization. The interactions between agents and their environment, which is based on the artificial potential fields approach, allow us to locally optimize the agent's location. The optimization of the whole system is then obtained from a self-organization of the agents. The efficiency of the proposed approach is confirmed by computational results based on a set of comparisons with the k-means clustering technique.

1 INTRODUCTION

The facility positioning problems have witnessed an explosive growth in the last four decades. As Krarup and Pruzan [8] point out, this is not at all surprising since location policy is one of the most profitable areas of applied systems analysis. This is due to the importance of location decisions which are often made at all levels of human organization. Then, such decisions are frequently strategic since they have consequential economic effects.

The term facility is used in its broadest sense. It refers to entities such as bus-stops, schools, hospitals, fire stations, etc. The general problem is, then, the location of new facilities to optimize some objectives such as distance, travel time or cost and demand satisfaction.

However, positioning problems are often extremely difficult to solve, at least optimally (often classified as NP-Hard). There have been works based on genetic algorithms, branch and bound, greedy heuristics, etc. These approaches are not easily adapted for dynamic systems where the system constraints or data change. This is a real limitation since most of real problems are subject to change and dynamics. To deal with this lack of flexibility and robustness, we adopt a multiagent approach which is known to be well suited for dynamical problems [5].

This paper proposes a multiagent approach for the facility location problem, which is based on the self-organization of reactive agents. To our knowledge, no reactive agent-based approaches have been already used to deal with this problem. The choice of a multiagent

approach provides several advantages. First, multiagent systems are well suited to model distributed problems. In such systems, several entities evolving/moving in a common environment have to cooperate to perform collective and local goals. Second, even if the multiagent approach does not guarantee to find optimal solution, it is, often, able to find satisfying ones without too much computational cost [19]. Through this paper we show that the reactive multiagent approach can be an interesting new way for optimization in positioning problems. Then, it provides satisfying solutions in addition to other assets as flexibility, modularity and adaptability to open systems. In our approach, agent behavior is based on the combination of attractive and repulsive forces. The idea is that the behavior of agents at the microscopic level leads to emergence of solutions at the macroscopic level [12].

This paper is structured as follows: section 2 presents the facility location problems. Then, section 3 details the proposed multiagent approach. Section 4 presents experimental evaluations through comparisons with the k-means approach. In section 5, some aspects of the model are discussed. Then, the last section gives some conclusions and perspectives.

2 THE POSITIONING PROBLEM

2.1 Overview

In the literature, the general facility positioning problem consists in locating new facilities to optimize some objectives such as distance, demand covering, travel time, cost.

There are four components that characterize location problems [14]: (1) a space in which demands and facilities are located, (2) a metric that indicates distance (or other measures as time) between demands and facilities, (3) demands, which must be assigned to facilities, and (4) facilities that have to be located. There exists two types of location problems: continuous and discrete ones. The problem is continuous when the facilities to be sited can generally be placed anywhere on the plane or on the network. In discrete location problems the facilities can be placed only at a limited number of eligible points.

A non-exhaustive list of facilities problems includes: p-center, p-median, set covering, maximal covering, dynamic location, stochastic location and multiobjective location problems. This paper focuses, particularly, on the p-median problem. The mathematical formulations of the previous variants are well known. However, formulating is only one step of analyzing a location problem. The other step and the most challenging one is to find optimal solutions.

Typically, the possible approaches to such a problem and especially to the p-median problem, consist in exact methods which allow to find optimal solutions. A well-known example of methods is

¹ Laboratory S.e.T, University of Technology Belfort Montbéliard 90010 Belfort, France, email: (sana.moujahed, olivier.simonin, abder.koukam)@utbm.fr

² Laboratory S.O.I.E, National School of computer science of Tunis 2010 La Manouba, Tunisia, email: khaled.ghedira@isg.rnu.tn

³ Deployment, location and siting are used as synonyms

branch and bound [18]. However, these solutions are quickly inefficient for very complex problems, i.e. with hundreds of constraints and variables. Then obtaining optimal solutions for these problems requires colossal computational resources.

Another category of methods are proposed for the p-median problem. These methods, known as heuristics, allow to find good solutions, but do not guarantee finding the optimal one(s): greedy heuristics [2], genetic algorithms [6], lagrangean relaxation [3], etc. However, these approaches have several drawbacks such as the rigidity, the lack of robustness and flexibility, the computational cost (huge population size and long convergence time, for example in genetic algorithms). Particularly, these approaches are limited in their ability to cope with dynamic problems characterized by the change of problem constraints and optimization criteria.

This paper explores another possible heuristic which is based on multiagent systems. The remainder of the paper will focus on this approach. The next section presents formally the continuous p-median problem.

2.2 Continuous p-median problem statement

In the rest of the paper, the continuous p-median problem is considered. It consists to locate a fixed number of facilities such that the whole environment can be used. The objective is to minimize the distance between demands and facilities.

The problem is expressed as follow [13]:

E = the set of demand points in the plane \mathbb{R}^2 (or more generally \mathbb{R}^n) indexed by e

W_e = a positive weight assigned to each demand

p = the maximum number of facility to locate

$d(x, e)$ = the distance between the facility x and the demand e

The problem is to find a subset X of p facility locations within a feasible region $S \subset \mathbb{R}^2$, such that:

$$\min_{X \subset S, |X|=p} F_E(X) \quad (1)$$

$$F_E(X) = \sum_{e \in E} W_e \cdot \min_{x \in X} d(x, e)$$

The objective function (1) minimizes the weighted sum of distances of the demand points to their closest facility.

3 A SELF-ORGANIZATION APPROACH FOR THE CONTINUOUS P-MEDIAN PROBLEM

Our model relies on the Artificial Potential Fields (APF) approach which is a possible manner to build self-organized systems. This approach is presented in the next section, the proposed model is detailed in section 3.2.

3.1 The artificial potential fields approach

Self-organization exists in many natural systems and especially in insect societies. Such systems are composed of simple entities, for instance ants, which can build tri-dimensional structures or solve complex problems without any global control [11]. Their organization results from the numerous interactions between agents and their environment. It is the environment that guides the agent behaviors and the whole system organization (called stigmergy principle) [12].

Such an approach has been used to define decentralized algorithm to deal with path finding problems (ant algorithm [4]), collective tasks (such as boxpushing [1], navigation [15], foraging with robots),

etc. Most of these works are based either on digital pheromones (as inspired by ants) or on artificial potential fields (APF). We adopt this second one because it is well suited to deal with spatial constraints, as it is the case in the p-median problem.

This APF approach has several inspirations (physical, biological, etc). The concept was introduced in Lewin's topological psychology [9]. The basic idea is that human behavior is controlled by a force field generated by objects or situations with positive or negative values or valences. During the past decade, potential field theory has gained popularity among researchers in the field of autonomous robots [7] and especially in robot motion planning thanks to their capability to act in continuous domains in real-time. By assigning repulsive force fields to obstacles and an attractive force field to the desired destination, a robot can follow a collision-free path via the computation of a motion vector from the superposed force fields [1]. In [16], artificial potential fields are used to tackle cooperation and conflict resolution between situated reactive agents.

However, the APF technique is limited by a well known drawback: local minima. Indeed, adding attractive and repulsive fields can produce areas where forces are equilibrated. Then, an agent that uses potential fields to move can be trapped in such places. The originality of our approach relies on the fact that we do not try to avoid such local minima. At the opposite, we exploit them as interesting places where facilities are located at the balance of different influences.

3.2 A self-organizing multiagent model

As facilities are elements to be placed in the environment, we model them as reactive agents. The environment is defined by a finite and continuous space. Demands, which are static data of the problem, are defined as an environment characteristic.

As in the reactive multiagent methodology proposed in [17], our approach consists to, first, define the behavior of a single agent (facility) so as to optimize its position considering the perceived demand. Second, we consider interactions between agents, to obtain the collective problem solving.

3.2.1 Local demand satisfaction

We first define the behavior of an agent which must minimize its distance to the perceived demand. The key idea is that demand induces attraction forces which are applied on the agent. Considering one demand point, an attractive force is defined from the agent towards the demand. It is expressed as a vector the intensity of which is proportional to the demand weight and to the distance between the agent and the demand. Formally, for an agent A perceiving a demand D with weight W_D

$$\vec{F}_{D/A} = W_D \cdot \overrightarrow{AD} \quad (2)$$

The influence of the attraction decreases when the agent moves towards the demand. Thus, if the agent attains the demand the attraction behavior is inhibited.

For the set of perceived demands, the influence on an agent is defined as the sum of all induced forces. Formally, the local attraction force undergone by an agent A is computed as follows:

$$\vec{F}_{demands/A} = \frac{\sum_{i=1}^n \vec{F}_{i/A}}{n} \quad (3)$$

n is the number of demands perceived by the agent A through its attraction radius r_a ($n = 5$ in Fig.1). The demand is indexed by i .

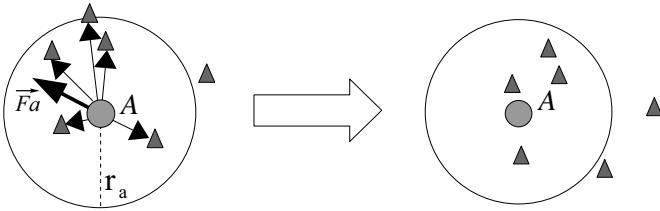


Figure 1. Attractions lead the agent to the weighted barycenter of demands

As a consequence the agent moves to the weighted barycenter of the demands, which is known to approach the minimum average distance to several close weighted points [10, 13]. For example, if an agent is subject to two attractive forces (from two different demands), it will be more attracted towards the biggest demand. Then, it will move towards a balance point. This point is defined as the place where the two attraction forces are equilibrated.

Now, we have to consider several agents applying such a behavior. Then, some of them could move to the same locations. In such a case the process is sub-optimal since several agents cover the same demand. To prevent such a process repulsive forces are introduced to the model.

3.2.2 Local coordination

In order to avoid that agents have the same locations, we introduce repulsive forces between them. It concerns close agents, i.e. situated under a particular distance, defined as the repulsion radius (r_r in Fig.2).

The force intensity is defined as inversely proportional to the inter-agent distance (see Fig.2).

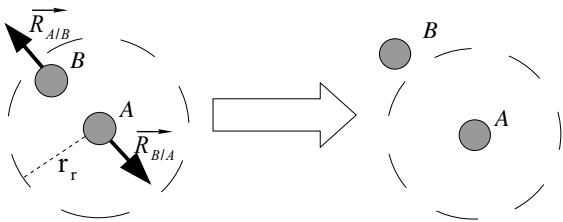


Figure 2. Repulsions between agents A and B lead them to keep away

Formally the repulsive force induced by an agent B on an agent A is expressed as follow:

$$\vec{R}_{B/A} = \frac{\vec{BA}}{\|\vec{AB}\|^2} \quad (4)$$

Then, the local repulsive force undergone by an agent A is computed as follows:

$$\vec{R}_{agents/A} = \frac{\sum_{j=1}^m \vec{R}_{j/A}}{m} \quad (5)$$

m is the number of agents perceived by the agent A. These agents are indexed by j . Fig.2 illustrates this repulsive process between two agents.

This repulsion process allows the coordination of agents while moving to the demand (see next section). Moreover, such repulsive forces can allow to respect constraints on minimal distances separating facilities (constraint present in many facility location applications).

3.2.3 Collective solving

The agent behavior is defined as the weighted sum of both local attraction and repulsion forces. Formally, for an agent A, it is expressed as follows:

$$\vec{Move} = \alpha \vec{F}_{demands/A} + (1 - \alpha) \vec{R}_{agents/A} \quad (6)$$

The coefficient α allows us to favour either the attraction or the repulsion.

We now consider the whole system, where several facilities must optimize their positioning to cover numerous demands. In the self-organizing approach, no global control is used. Agents are created and distributed in the environment and act following the defined individual behavior.

To implement the proposed multiagent model, we can (i) assign a thread to each agent or (ii) define a scheduler that simulates the parallel agents computation. We adopt the second solution which is generally used for reactive agents implementation. Finally, the collective solving process is presented in Algorithm 1. The initialization (step 1) and the fitness computation (step 9) are detailed in the next section.

Algorithm 1 Collective solving process

```

1: Initialization of Agent positions
2: while Fitness in progress do
3:   for all Agents do
4:     Attraction computation
5:     Repulsion computation
6:     Move computation
7:     Move execution
8:   end for
9:   Fitness computation
10: end while
```

4 EXPERIMENTATIONS

After exposing the principle of the approach, the model is evaluated on a case study. It consists in positioning facilities (bus-stops, restaurants, etc) on a continuous environment corresponding to the map of France presented in Fig.3 (400x400 size). It contains the demand weights which are values between 0 and 255 (randomly generated). These weights are represented as a gradation from black color (255) to white color (0).

The initial positioning of facilities is performed with a random computation (Fig.3 (a)). Parameters values are: $\alpha = 0.5$, $r_a = 25$, $r_r = 20$.

When the algorithm starts, facility agents (the white points in Fig.3) move towards demands while avoiding other agents (Fig.3 (b)). In the first iterations we can observe important moves to the highest demands while the repulsive forces globally stabilize this tendency. The system iterates until it attains a global equilibrium state (convergence to a stable state).

Fig.3 (c) shows the final state to which the system converges. The facilities repartition is characterized by an intensification of the

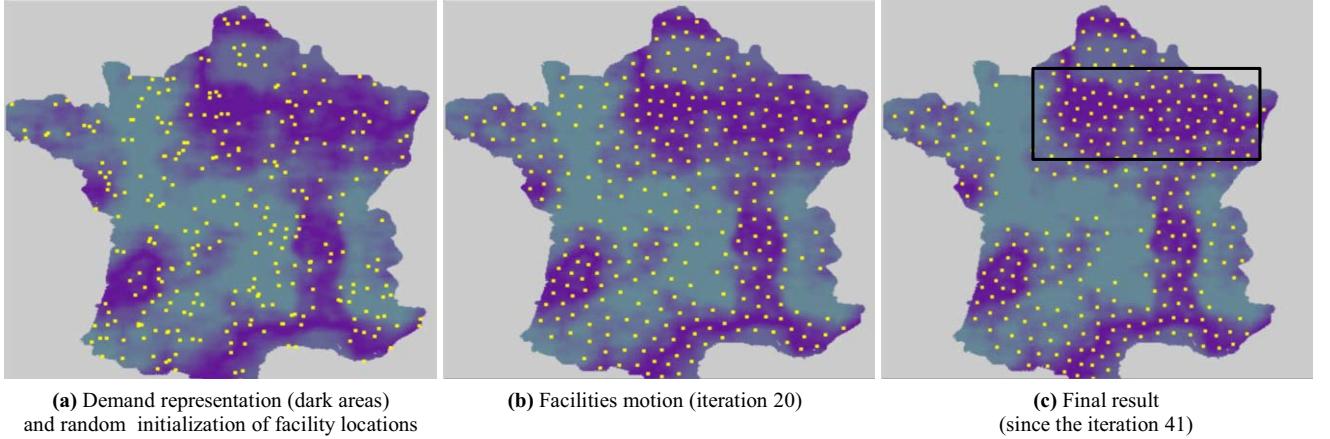


Figure 3. The evolution of facilities positioning for the case study with 400 agents

agents in areas where demands is high. This result is visible inside the rectangular area. It is also clear that all facilities respect a minimal distance between them.

The performance of the multiagent model has been compared with the k-means clustering technique. The k-means algorithm is a well known technique that computes very good solutions to the facility location problem [10]. It allows us to classify or to group objects based on attributes/features into K number of groups. The grouping is done by minimizing the sum of distances between data and the corresponding cluster centroid (Algorithm 2).

Algorithm 2 The k-means clustering

-
- 1: **repeat**
 - 2: Place k points into the space represented by the objects that are being clustered.
 - 3: Assign each object to the group that has the closest centroid. When all objects have been assigned, recalculate the positions of the k centroids as weighted barycenters.
 - 4: **until** The centroids no longer move.
-

Comparisons are made according to a global fitness index expressed by the formula (7) and corresponding to the mean distance between each demand and the nearest facility:

$$Fitness = \frac{\sum_{ij} D_{ij} * d(C_{ij}, x_{ij})}{\sum_{ij} D_{ij}} \quad (7)$$

D_{ij} = the demand at point x_{ij}

$d(C_{ij}, x_{ij})$ = the distance between the point x_{ij} and the nearest facility C_{ij}

Comparisons are carried out on different number of facilities, as shown in Table 1. For each facility number, 50 tests have been executed. The fitness values obtained by applying the multiagent approach are very close to the k-means ones. The difference is small and it is inversely proportional to the number of facilities.

A second comparison has been performed considering another criterion: the time required to converge to a solution. Results are presented in Table 2, they show that the multiagent model converges to a solution more rapidly than the k-mean, e.g. for 400 facilities the multiagent approach is 3 times faster than the k-means. It is particularly

Table 1. Comparison with k-means clustering

Fitness: minimal values					
Facilities	50	100	150	200	400
Multiagent	16,592	11,187	9,164	7,945	5,696
k-means	15,556	10,965	9,010	7,820	5,593
Difference	1,036	0,222	0,154	0,125	0,095

interesting to note that the multiagent approach is more efficient than the k-means while the number of agents increases.

Table 2. Comparison of computation time

Computation time (in second)			
Facilities	100	150	400
Multiagent	34.289	24.925	33.019
k-means	37.678	73.306	118.251

Fig.4 plots the evolution of the fitness values for 400 facilities. We can show that the fitness decreases until the convergence to a constant value. Here, the convergence is attained rapidly: since the 41 th iteration.

All the experimentations have shown that the agents systematically converge to a stable location. It corresponds to a global balance between attraction to demands and inter-agents repulsive forces.

5 DISCUSSION

The previous experimentations allow to point up some observations on the proposed model. The obtained solutions are globally satisfying considering the fitness values. We have shown that these solutions are quickly obtained.

For each specific application, the multiagent approach needs a parameter setting stage. However, the proposed model depends only on three parameters: attraction and repulsion radius, and the weight combination of influences (α in formula (6)). Attraction and repulsion radius depend on the considered application. Generally, the attraction radius is defined by the coverage distance (for a demand, the next facility must be within a specified distance, the covering radius).

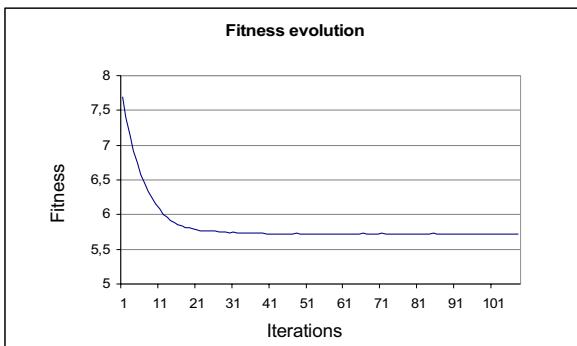


Figure 4. The fitness evolution for the case study with 400 agents

The repulsion radius is defined by the minimal distance allowed between two facilities. α is defined following the designer objectives.

Generally, the existing solutions for facility positioning [2, 18, 6] are not easily adaptable when the problem constraints change, particularly, for dynamic perturbations. It can concern the environment structure (e.g. demands), the facilities number, etc. For instance, when facilities have been located, any change in the demand will generally needs a new execution of the employed algorithm (it is the case with genetic algorithms, branch and bound, etc). At the opposite, our self-organizing approach will immediately adapt the locations to the perturbation. The algorithm can compute this adaptation from its last state. Moreover, it is interesting to not stop the algorithm in order to observe new solutions which are dynamically obtained while perturbing the system.

The proposed model can be also adapted to variations in the problem statement. For instance, we have applied our model to bus-stops positioning in a real bus-line network. The considered network serves a 60000 inhabitants city. Demands correspond to real values of inhabitants density per quarter. In this problem the positioning of facilities is limited to the lines. The model has been adapted to this new constraint without changing the agent behaviors. We have just constrained the agents to stay on lines (the move vector is transformed so as the agents move along lines). Experiments have shown that the fitness value decreases until its convergence to a constant value, which corresponds to an optimization of the bus-stops location.

6 CONCLUSIONS

This paper has presented a self-organizing multiagent approach for the continuous p-median problem. Facilities, which are modeled as reactive agents, move according to their local perception. Demands induce attractive forces and the proximity between agents generates repulsive ones. The agent behavior is defined as a combination of these two kind of opposite influences.

Local minima, which must be avoided in the artificial potential fields approach are exploited in our model as balance points between demands.

At a collective level, we have shown that the system, i.e. the whole set of agents, interact to minimize distances to demands. Then, the system converges to a global stable state.

The relevance of our approach has been shown through its application to location of facilities on a continuous environment. In particular, it has been compared to the k-means clustering algorithm. A first

evaluation criterion concerns the variation of the final fitness value. The multiagent results tend to the k-means values when the number of agents increases. The second evaluation concerns the computation time to obtain a stable solution. In this case, the multiagent approach is clearly faster and this advantage grows with the number of agents. These different evaluations show that a self-organizing multiagent approach can be an interesting perspective to optimization in positioning problems.

Future works deal, first, with a more formal evaluation of the global system convergence. Then, we seek to apply our approach to another problematic in location problems: the dimensioning problem. It consists to optimize the number of facilities to locate, since each new facility increases the location cost. We obtain a multicriteria problem. We then propose to add two behaviors allowing the creation and the removing of agents in order to optimize facilities location and number.

REFERENCES

- [1] R C. Arkin, *Behavior-Based Robotics*, MIT Press, 1998.
- [2] V. Chvatal, 'A greedy heuristic for the set covering problem', *Mathematics of Operations Research*, **4**(3), 233–235, (1979).
- [3] M S. Daskin, *Network and discrete location: Models, Algorithms, and applications*, John Wiley, New York, 1995.
- [4] M. Dorigo and C. Blum, 'Ant colony optimization theory: A survey', *Theoretical Computer Science*, **344**(2-3), 243–278, (2005).
- [5] J. Ferber, *Multi-agent Systems : An Introduction to Distributed Artificial Intelligence*, Addison Wesley, 1999. ISBN: 0-201-36048-9.
- [6] J H. Jaramillo, J. Bhadury, and R. Batta, 'On the use of genetic algorithms to solve location problems', *Computers & Operations Research*, **29**, 761–779, (2002).
- [7] O. Khatib, 'Real-time obstacle avoidance for manipulators and mobile robots', in *IEEE international conference on robotics and automation*, pp. 500–505, (1985).
- [8] J. Krarup and P M. Pruzan, 'The simple plant location problem: Survey and synthesis', *European Journal of Operations Research*, **12**, 36–81, (1983).
- [9] K. Lewin, *Principles of topological psychology*, McGraw-Hill Book Company, Inc, 1936.
- [10] A. Likas, N A. Vlassis, and J J. Verbeek, 'The global k-means clustering algorithm', *Pattern Recognition*, **36**(2), 451–461, (2003).
- [11] H V D. Parunak, 'Go to the ant: Engineering principles from natural multi-agent systems', *Annals of Operations Research*, **75**, 69–101, (1997).
- [12] H V D. Parunak and S A. Brueckner, 'Entropy and self-organization in multi-agent systems', in *Fifth International Conference on Autonomous Agents*, pp. 124–130, (2001).
- [13] F. Plastria, 'On the choice of aggregation points for continuous p-median problems: a case for the gravity centre', *TOP*, **9**, 217–242, (2001).
- [14] C S. ReVelle and H A. Eiselt, 'Location analysis: A synthesis and survey', *European Journal of Operations Research*, **165**, 1–19, (2005).
- [15] C W. Reynolds, 'Flocks, herds, and schools: A distributed behavioral model', *Computer Graphics*, **21**(4), 25–34, (July 1987).
- [16] O. Simonin and J. Ferber, 'Modeling self satisfaction and altruism to handle action selection and reactive cooperation', in *Sixth International Conference on the Simulation of Adaptive Behavior*, pp. 314–323, Paris, France, (2000).
- [17] O. Simonin and F. Gechter, 'An environment-based principle to design reactive multi-agent systems for problem solving', *LNAI, Environments for Multiagent Systems II*, **3830**, 32–49, (2006).
- [18] D. Tcha and B. Lee, 'A branch-and-bound algorithm for the multi-level uncapacitated location problem', *European Journal of Operations Research*, **18**, 35–43, (1984).
- [19] M. Wooldridge, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, chapter 1 Intelligent Agent, 27–77, G. Weiss, 1999.

Arguing with Confidential Information

Nir Oren and **Timothy J. Norman** and **Alun Preece¹**

Abstract. While researchers have looked at many aspects of argumentation, an area often neglected is that of argumentation strategies. That is, given multiple possible arguments that an agent can put forth, which should be selected in what circumstances. In this paper, we propose a heuristic that implements one such strategy. The heuristic assigns a utility cost to revealing information, as well as a utility to winning, drawing and losing an argument. An agent participating in a dialogue then attempts to maximise its utility. We present a formal argumentation framework in which this heuristic may operate, and show how it functions within the framework. Finally, we discuss how this heuristic may be extended in future work, and its relevance to argumentation theory in general.

1 Introduction

Argumentation has emerged as a powerful reasoning mechanism in many domains. One common dialogue goal is to persuade, where one or more participants attempt to convince the others of their point of view. This type of dialogue can be found in many areas including distributed planning and conflict resolution, education and in models of legal argument. At the same time that the breadth of applications of argumentation has expanded, so has the sophistication of formal models designed to capture the characteristics of the domain. Prakken [11] for example, has focused on legal argumentation, and has identified four layers with which such an argumentation framework must concern itself. These are:

- The logical layer, which allows one to represent basic concepts such as facts about the world. Most commonly, this layer consists of some form of non-monotonic logic.
- The dialectic layer, in which argument specific concepts such as the ability of an argument to defeat another are represented.
- The procedural layer governs the way in which argument takes place. Commonly, a dialogue game [17] is used to allow agents to interact with each other.
- The heuristic layer contains the remaining parts of the system. Depending on the form of the underlying layers, these may include methods for deciding which arguments to put forth and techniques for adjudicating arguments.

While many researchers have focused on the lowest two levels (excellent surveys can be found in [3, 11, 12]), and investigation into various aspects of the procedural layer is ongoing (for example, [16, 6]), many open questions remain at the heuristic level.

In this paper, we propose a decision heuristic for an agent allowing it to decide which argument to put forth. The basis for our idea is simple; the agent treats some parts of its knowledge as more confidential than other parts, and, while attempting to win the argument, attempts

to reveal as little of the more secret information to others as possible. This concept often appears in adversarial argumentation under the guise of not mentioning information that might help an opponent's case, but also appears in many other settings. For example, revealing trade secrets, even to win an argument, may damage an agent in the long term. The heuristic often emerges in negotiation dialogues, as well as persuasion dialogues in hostile setting (such as takeover talks or in some legal cases). Utilising this heuristic in arguments between computer agents can also be useful; revealing confidential information in an ongoing dialogue may damage an agent's chances of winning a future argument.

In the next section, we examine a few existing approaches to strategy selection, after which we discuss the theoretical foundations of our approach. We then present the heuristic, after which we see how it operates by means of an example. We conclude the paper by looking at possible directions in which this work can be extended.

2 Background and related research

Argumentation researchers have recognised the need for argument selection strategies for a long time. However, the field has only recently started receiving more attention. Moore, in his work with the DC dialectical system [7], suggested that an agent's argumentation strategy should take three things into account:

- Maintaining the focus of the dispute.
- Building its point of view or attacking the opponent's one.
- Selecting an argument that fulfils the previous two objectives.

The first two items correspond to the military concept of a strategy, i.e. a high level direction and goals for the argumentation process. The third item corresponds to an agent's tactics. Tactics allow an agent to select a concrete action that fulfils its higher level goals. While Moore's work focused on natural language argument, these requirements formed the basis of most other research into agent argumentation strategies.

In 2002, Amgoud and Maudet [1] proposed a computational system which would capture some of the heuristics for argumentation suggested by Moore. Their system requires very little from the argumentation framework. A preference ordering is needed over all possible arguments, and a level of prudence is assigned to each agent. An argument is assigned a strength based on how convoluted a chain of arguments is required to defend it. An agent can then have a "build" or "destroy" strategy. When using the build strategy, an agent asserts arguments with a strength below its prudence level. If it cannot build, it switches to a destroy strategy. In this mode, it attacks an opponent's arguments when it can. While the authors note other strategies are reasonable, they make no mention of them. Shortcomings of their approach include its basis on classical propositional logic and the assumption of unbounded rationality; computational limits may affect

¹ Department of Computing Science, University of Aberdeen, AB24 3UE, Scotland, email: noren,tnorman,apreece@csd.abdn.ac.uk

the arguments agents decide to put forth. Finally, no attempt is made to capture the intuition that a fact defended by multiple arguments is more acceptable than one defended by fewer (the so called “accrual of evidence” argument scheme [9]).

Using some ideas from Amgoud’s work, Kakas et al. [5] proposed a three layer system for agent strategies in argumentation. The first layer contains “default” rules, of the form $\text{utterance} \leftarrow \text{condition}$, while the two higher layers provide preference orderings over the rules. Assuming certain restrictions on the rules, they show that only one utterance will be selected using their system, a trait they refer to as determinism. While their approach is able to represent strategies proposed by a number of other techniques, it does require hand crafting of the rules. No suggestions are made regarding what a “good” set of rules would be.

In [2], Bench-Capon describes a dialogue game based on Toulmin’s work. He identifies a number of stages in the dialogue in which an agent might be faced with a choice, and provides some heuristics as to what argument should be advanced in each of these cases. Only an informal justification for his heuristics is provided.

3 Confidentiality Based Argumentation

In many realms of argument, auxiliary considerations (apart from simply winning or losing the argument) come into play. In many scenarios, one such consideration involves hiding certain information from an opponent. In this section, we describe a utility based heuristic to guide an agent taking part in a dialogue while being careful about what information it reveals. When faced with a number of possible arguments that it can advance, we claim it should put forth the one that minimises the exposure of information that it would like to keep private. The limitations of our current approach, as well as extensions and refinements to it are discussed in Section 5.

Our formalism is based on many ideas from other formal argument systems (e.g. [4, 10, 8, 15]). We use our own formal argumentation system which introduces features not seen in others, and which allows us to study our proposed heuristic in isolation. Our argumentation framework is very simple, and does not contain features such as a preference ordering on arguments which allows one to overrule another. Our system is formalised in two parts. After specifying the argumentation framework, our heuristic is described, in terms of the framework.

3.1 The Argumentation Framework

Argumentation takes place over the language Σ , which contains propositional literals and their negation.

Definition 1 Argument An argument is a pair (P, c) , where $P \subseteq \Sigma \cup \{\top\}$ and $c \in \Sigma$ such that if $x \in P$ then $\neg x \notin P$. We define $\text{Args}(\Sigma)$ to be the set of all possible arguments derivable from our language.

P represents the premises of an argument (also referred to as an argument’s support), while c stands for an argument’s conclusion. Informally, we can read an argument as stating “if the conjunction of its premises holds, the conclusion holds”. Facts can be represented using the form (\top, a) .

Arguments interact by supporting and attacking each other. Informally, when an argument attacks another, it renders the latter’s conclusions invalid.

Definition 2 Attack An argument $A = (P_a, c_a)$ attacks $B = (P_b, c_b)$ if $\exists f \in P_b$ such that $f = \neg c_a$ or $c_a = \neg c_b$. For convenience, we write this as $\text{attacks}(A, B)$.

An argument is only relevant to an instance of argumentation if its premises are true. We call such an argument *justified*. However, a simple definition of this concept can cause problems when it comes to self attacking (or self defending) arguments, as well as circular reasoning, and care must thus be taken when describing this concept. Before doing so, we must (informally) describe the proof theory used to determine which literals and arguments are in effect at any time.

The idea behind determining what arguments and literals are admissible at any time is as follows. We start by looking at the facts, and determining what knowledge can be derived from them by following chains of argument. Whenever a conflict occurs (i.e. we are able to derive both x and $\neg x$), we remove these literals from our derived set. Care must be taken to also get rid of any arguments (and further facts) derived from any conflicting literals. To do this, we keep track of the conflicting literals in a separate set, whenever a new conflict arises, we begin the knowledge determination process afresh, never adding any arguments whose conclusions are in the conflicting set to the knowledge set. The philosophical and practical ramifications of this approach are examined in Section 5.

More formally, an instance of the framework creates two sets $J \subseteq \text{Args}(\Sigma)$ and $C \subseteq \Sigma$ representing justified arguments and conflicts respectively.

Definition 3 Derivation An argument $A = (P_a, c_a)$ is derivable from a set S given a conflict set C (written $S, C \vdash A$) iff $c_a \notin C$ and $(\forall p \in P_a : (\exists s \in S \text{ such that } s = (P_s, p) \text{ and } p \notin C) \text{ or } P_a = \{\top\})$.

Clearly, we need to know what elements are in C . Given a knowledge base of arguments $\kappa \subseteq \text{Args}(\Sigma)$, this can be done with the following reasoning procedure:

$$\begin{aligned} J_0 &= \{A | A \in \kappa \text{ such that } \{\}, \{\} \vdash A\} \\ C_0 &= \{\} \end{aligned}$$

Then, for $i > 0, j = 1 \dots i$, we have:

$$C_i = C_{i-1} \cup \{c_A, \neg c_A | \exists A = (P_A, c_A), B = (P_B, \neg c_A) \in J_{i-1} \text{ such that } \text{attacks}(A, B)\}$$

$$\begin{aligned} X_{i0} &= \{A | A \in \kappa \text{ and } \{\}, C_i \vdash A\} \\ X_{ij} &= \{A | A \in \kappa \text{ and } X_{i(j-1)}, C_i \vdash A\} \end{aligned}$$

$$J_i = X_{ii}$$

The set X allows us to recompute all derivable arguments from scratch after every increment of i^2 . Since i represents the length of a chain of arguments, when $i = j$ our set will be consistent to the depth of our reasoning, and we may assign all of these arguments to J . Eventually, $J_i = J_{i-1}$ (and $C_i = C_{i-1}$) which means there are no further arguments to find. We can thus define the conclusions asserted by κ as $K = \{c | A = (P, c) \in J_i\}$, for the smallest i such that $J_i = J_{i+1}$. We will use the shorthand $K(\kappa)$ and $C(\kappa)$

² This allows us to get rid of long invalid chains of arguments, as well as detect and eliminate arbitrary loops

to represent those literals which are respectively asserted by, or in conflict with the knowledge base κ .

We provide an example which illustrates the approach (note that not all steps are shown):

Example 1 $\kappa = \{(\top, s), (s, t), (t, \neg s)\}$

$$J_0 = \{(\top, s)\}, C_1 = \{\}$$

$$J_1 = X_{11} = \{(\top, s), (s, t)\}$$

...

$$J_2 = (\top, s), (s, t), (t, \neg s), C_3 = \{s, \neg s\}$$

$$X_{30} = \{\} \dots J_4 = J_3 = \{\}$$

3.2 The Heuristic

Agents engage in a dialogue using the argumentation framework described above in an attempt to persuade each other of certain facts. An agent has a private knowledge base (KB) as well as a goal literal g and a preference ranking ρ which specifies an agent's reluctance to reveal certain literals. The environment, apart from containing agents, contains a public knowledge base which takes on a role similar to a global commitment store, and we thus refer to it as CS below.

Definition 4 Environment An environment is a pair $(Agents, CS)$ where $Agents$ is the set of agents participating in the dialogue and $CS \subseteq \text{Args}(\Sigma)$

Definition 5 Agent An Agent $\alpha \in Agents$ is a tuple $(Name, KB, \rho, g, U_{win}, U_{draw}, U_{lose})$ where $KB \subseteq \text{Args}(\Sigma)$, $g \in \Sigma$. ρ is a preference ranking function and $U_{win}, U_{draw}, U_{lose} \in \mathbb{R}$ are the utilities gained for winning, drawing or losing an argument.

The preference ranking expresses the “cost” to an agent of revealing certain information in a specific context. It maps a set of literals L to a real number. The cost of being in a certain environmental state is the result of applying the preference ranking function ρ to the literals present in that state.

Definition 6 Preference Ranking A preference ranking ρ is a function $\rho : L \rightarrow \mathbb{R}$ where $L \subseteq 2^\Sigma$.

Agents take turns to put forward a line of argument consisting of a number of individual arguments. For example, an agent could make the utterance $\{(\top, a), (a, b)\}$. Alternatively, an agent may pass (by uttering an empty argument $\{\}$). The dialogue ends when CS has remained unchanged for n turns i.e. after all players have had a chance to modify it, but didn't (this is normally caused by all agents having passed consecutively). Once this has happened, the acceptable set of arguments is computed over the CS , and the status of each agent's goal can be determined, allowing one to compute the winners of the game.

Definition 7 Turns and utterances The function

$$\text{turn} : \text{Environment} \times \text{Name} \rightarrow \text{Environment}$$

takes an environment (of the form $\text{Environment} = (Agents, CS)$) and an agent label, and returns a new environment containing the result of the utterance ($\text{utterance} : \text{Environment} \times \text{Name} \rightarrow 2^{\text{Args}(\Sigma)}$) made by the labelled agent during its turn.

$$\text{turn}(\text{Environment}, \alpha) = (\text{Agents}, \{CS \cup \text{utterance}(\text{Environment}, \alpha)\})$$

At turn i , we set $\alpha = \text{Agent}_i \bmod n$, where n is the number of agents taking part in the dialogue. The *utterance* function is dependent on the agent's strategy, and we will describe one such strategy below. Before doing so, we define the dialogue game itself. Each turn in the dialogue game results in a new public commitment store, which is used by agents in later turns.

Definition 8 Dialogue game The dialogue game is defined as
 $turn_0 = \text{turn}((\text{Agents}, CS_0), \text{Agent}_0)$
 $turn_{i+1} = \text{turn}(turn_i, \text{Agent}_i \bmod n)$

The game ends when $turn_i \dots turn_{i-n+1} = turn_{i-n}$.

CS_0 is dependent on the system, and contains any arguments that are deemed to be common knowledge (though these arguments may be attacked like any other argument during later turns in the game). Also, note that the null utterance $\{\}$ is defined to be a pass.

By using the procedure described earlier, agents can

- Determine, by looking at CS , what literals are in force (i.e. in $K(CS)$) and in conflict.
- Determine, by combining CS with parts of their own knowledge base, what literals they can prove (or cause to conflict).

By doing the latter, together with examining which literals are introduced into K and C , as well as their cost as computed from ρ , an agent will narrow down the range of arguments it will consider submitting, though it may still have multiple arguments to choose from. It should be noted that an agent might be willing to draw or even lose an argument rather than reveal too much information. Winning (or drawing) an argument earns the agent a certain amount of utility. Thus, the final choice about which argument to put forth is based on the effect of the argument in combination with its utility cost. We begin by defining the set of winning and drawing arguments paying no attention to the argument cost.

Definition 9 Winning arguments An agent's set of winning arguments is defined as
 $Win = \{A \in 2^{KB} \mid g \in K(A \cup KB) \text{ and if } A \neq \{\}, \{\} \notin A\}$.

Definition 10 Drawing arguments An agent's set of drawing arguments is defined as

$$Draw = \{A \in 2^{KB} \mid (g \in C(A \cup KB) \text{ or } \{g, \neg g\} \not\subseteq K(A \cup KB)) \text{ and if } A \neq \{\}, \{\} \notin A\}$$

While many possibilities exist as to how to weigh the cost of information, for reasons discussed in Section 5, we compute the information cost based on the literals present in CS after an argument has been advanced. Currently we make no distinction between whether information is revealed due to an utterance we make, or whether another agent revealed it.

Definition 11 Argument utility Given an agent with a preference ranking ρ , we define an agent's net utility U for advancing an argument A as

$$U(A) = \begin{cases} U_{win} - \rho(L) & \text{if } A \in Win \\ U_{draw} - \rho(L) & \text{if } A \in Draw \\ U_{lose} - \rho(L) & \text{otherwise} \end{cases}$$

such that $L = K(CS \cup A) \cup C(CS \cup A)$

The utterance an agent makes is chosen from the set of arguments that maximise its utility:

$$\text{utterance} \in \{a \subseteq A \mid \forall a, b \ U(a) \geq U(b)\}$$

At the end of the game, the literals in $K(CS)$ will be those for which undefeated arguments exist.

4 Example

To increase readability, we present our example in a somewhat informal manner. The argument consists of a hypothetical dialogue between a government and some other agent regarding the case for, or against, weapons of mass destruction (WMDs) existing at some location.

Assume that our agent ($Agent_0$) would like to show the existence of WMDs, i.e. $g = WMD$. Assume further that $U_{win} = 100$, $U_{draw} = 50$, $U_{lose} = 0$ and that the following arguments exist in the agent's private KB (where the context is clear, we omit brackets):

$(\top, spysat), (\top, chemicals), (\top, news), (\top, factories)$

$(\top, smuggling), (smuggling, \neg medicine), (news, WMD)$

$(\{factories, chemicals\}, WMD), (spysat, WMD)$

$(\{sanctions, smuggling, factories, chemicals\}, \neg medicine)$

While we will not fully describe the agents preference rating function ρ , we set the costs for tuples including literals as follows:

$(spysat, 100)$	$(chemicals, 30)$
$(news, 0)$	$(\{medicine, chemicals\}, 50)$
$(smuggling, 30)$	$(factories, 0)$

Note that if both medicine and chemicals are present, the agent's utility cost is 50, not 80.

As an example, ρ of an environment state containing both *spysat* and *chemicals* will be assigned a cost of 130.

The dialogue might thus proceed as follows:

- (1) $Agent_0 : (\top, news), (news, WMD)$
- (2) $Agent_1 : (\top, \neg news)$
- (3) $Agent_0 : (\top, factories), (\top, chemicals), (\{factories, chemicals\}, WMD)$
- (4) $Agent_1 : (\top, sanctions), (\{sanctions, factories, chemicals\}, medicine), (medicine, \neg WMD)$
- (5) $Agent_0 : (\top, smuggling), (\{sanctions, smuggling, factories, chemicals\}, \neg medicine)$
- (6) $Agent_1 : \{\}$
- (7) $Agent_0 : \{\}$

Informally, the dialogue proceeds as follows: $Agent_0$ claims that WMDs exist since the news says they do. $Agent_1$ retorts that he has not seen those news reports. $Agent_0$ then points out that factories and chemicals exist, and that these were used to produce WMDs. In response, $Agent_1$ says that due to sanctions, these were actually used to produce medicine. $Agent_0$ attacks this argument by pointing out that smuggling exists, which means that the factories were not used to produce medicines, reinstating the WMD argument. Both agents have nothing more to say, and thus pass. $Agent_0$ thus wins the game.

It should be noted that while $Agent_0$ is aware that spy satellites have photographed the WMDs, it does not want to advance this argument due to the cost of revealing this information. The final utility gained by $Agent_0$ for winning the argument is 20: 100 for winning the argument, less 30 for revealing *smuggling*, and 50 for the presence of the *chemicals* and *medicine* literals. Also, note that the fact

that $Agent_1$ revealed the existence of medicines cost $Agent_0$ an additional 20 utility. This is somewhat counterintuitive, and extensions to overcome this behaviour are examined in the next section.

5 Discussion

This section examines the argumentation framework and the heuristic, tying it back to the concept of an argumentation strategy as proposed by Moore. We also examine some of the novel features of argument that emerge when dialogue takes place in the framework using the heuristic, and propose avenues for future research.

It should be noted that nothing in our framework forces literals to cost utility. Many scenarios can be imagined wherein revealing a literal causes a utility gain. For example, if an agent would like to steer a conversation towards a certain direction, it might gain utility for revealing literals relating to that topic, even though those might, in the long run, weaken its argument.

Our approach seems to share much in common with the "sceptical" approach to argumentation. When arguments conflict, we refuse to decide between them, instead ruling them both invalid. This means that our reasoning procedure is not complete, given the (rather convoluted) set of arguments $(\top, A), (\top, B), (A, \neg B), (B, \neg A), (A, C), (B, C), (\neg A, C), (\neg B, C)$ we can intuitively see that C should hold, but doesn't. Other argumentation systems (namely those utilising the unique-status-assignment approach [12]) are similarly incomplete, leaving this an open area for future research. Our sceptical approach does yield a sound system, as no conflicting arguments will remain in the final set of arguments.

The simplicity of our approach means that only specific types of arguments can be represented (namely, those whose premises are a conjunction of literals, and whose conclusion is a single literal). However, as seen in the example, even with this limitation, useful arguments can still emerge.

We developed our own argumentation framework rather than using an existing one for a number of reasons, including:

- The abstract nature of many frameworks (e.g. [4]) makes arguments atomic concepts. We needed a finer level of granularity so as to be able to talk about which facts are exposed (allowing us to measure the amount of information revealed during the dialogue process). Less abstract frameworks (e.g. [13, 10]), while looking at concepts such as derivability of arguments, still have as their main focus, the interactions between arguments.
- Almost all other frameworks define higher level concepts in terms of arguments attacking, defeating and defending one another. For us, the concept of one argument justifying another is critical, together with the concept of attack.
- Other argumentation systems contain concepts which we do not require, such as a preference ordering over arguments.

Another significant difference between our argumentation framework and most existing approaches is the scope of arguments. In our approach, agents can be aware of and utter arguments of which other agents are unaware. For example, even if no other agent knew of the literals X and Y , an agent could make the utterance $(\{X, Y\}, Z)$. An agent arguing for $\neg Z$ would then have no choice but to try obtain a draw result.

While representing the heuristic using one of the other approaches is (probably) not impossible, it appears to be more difficult than by using our own system.

Looking at Moore's three criteria for an agent argumentation strategy, we see that our heuristic fulfils its requirements. If the focus of

the argument were not maintained, more information would be given than is strictly necessary to win, thus fulfilling the first requirement. Both the second and third requirements are clearly met by the decision procedure for which argument to advance described in Definition 11.

The way in which we represent the information ‘leaked’ during the dialogue, as well as calculate the agent’s net utility, while simple, allows us to start studying dialogues in which agents attempt to hide information. Until now, most work involving utility and argumentation has focused on negotiation dialogues (e.g. [14]). We propose a number of possible extensions to the work presented in this paper.

One simple extension involves the addition of a context to the agent’s cost. In other words, given that fact A , B and C are known, we would like to be able to capture the notion that it is cheaper to reveal D and E together than as speech acts at different stages of the dialogue. Another form of context, which often appears in real world dialogues, occurs when two different pieces of information help derive a third, at different costs. In this case, the agent might be happy to reveal one without revealing the other, but currently, we are unable to perform complex reasoning about which to reveal. Without some form of lookahead to allow the agent to plan later moves, this extension is difficult to utilise. Once some form of lookahead exists, the addition of opponent modelling can further enhance the framework. Experimentally, evaluating the effects of various levels of lookahead, as well as different forms of opponent modelling might yield some interesting results.

Currently, we do not differentiate between information which the agent has explicitly committed to, and information that the agent has not yet disagreed with. More concretely, assume that the CS contains the argument (\top, A) . If an agent makes use of this argument, perhaps by submitting the argument (A, B) , then it is committed to the fact that A is true. If however, it never puts forth arguments making use of the fact, then an opponent cannot know if the agent is actually committed to A or not. We plan to extend our formalism and heuristic to capture this interaction in the near future.

Another extension that emerges from this line of reasoning is the concept of lying. An agent might commit to A to win an argument, even if its knowledge base contains only $\neg A$. How best to deal with this situation is an open question.

The way in which we handle conflicts is also open to debate. At the argumentation framework level, enhancements are required that allow one to present further evidence in support of a literal. By increasing the complexity of the model, methods for retracting literals can be introduced, opening up a whole host of questions at the heuristic level. For example, how does retracting support for a literal influence the information an opponent has of the retracting agent’s knowledge base?

6 Conclusions

In this paper, we proposed a heuristic for argumentation based on minimising the cost of information revealed to other dialogue participants. While such an argumentation strategy arises in many real world situations, we are not familiar with any application that explicitly makes use of this technique. To study the heuristic, we proposed an argumentation framework that allowed us to focus on it in detail. Several novel features emerged from the interplay between the heuristic and the framework, including the ability of an agent to win an argument that it should not have been able to win (if all information were available to all dialogue participants). While we have only examined a very abstract model utilising the heuristic, we believe

that many interesting extensions are possible.

Acknowledgements

This work is partly funded by the DTI/EPSRC E-Science Core Program and British Telecom, via a grant for the CONOISE-G project (<http://www.conoise.org>), a multi-university collaboration between Aberdeen, Cardiff and Southampton universities, and BT.

REFERENCES

- [1] Leila Amgoud and Nicolas Maudet, ‘Strategical considerations for argumentative agents (preliminary report)’, in *NMR*, pp. 399–407, (2002).
- [2] Trevor J.M Bench-Capon, ‘Specification and implementation of Toulmin dialogue game’, in *Proceedings of JURIX 98*, pp. 5–20, (1998).
- [3] Carlos Ivan Chesñevar and Ana Gabriela Maguitman, ‘Logical models of argument’, *ACM Computing Surveys*, **32**(4), 337–383, (2000).
- [4] Phan Minh Dung, ‘On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games’, *Artificial Intelligence*, **77**(2), 321–357, (1995).
- [5] Antonis C. Kakas, Nicolas Maudet, and Pavlos Moraitis, ‘Layered strategies and protocols for argumentation-based agent interaction’, in *ArgMAS*, pp. 64–77, (2004).
- [6] Peter McBurney and Simon Parsons, ‘Risk agoras: Dialectical argumentation for scientific reasoning’, in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 371–379, Stanford, USA, (2000).
- [7] David Moore, *Dialogue game theory for intelligent tutoring systems*, Ph.D. dissertation, Leeds Metropolitan University, 1993.
- [8] John L. Pollock, ‘Perceiving and reasoning about a changing world.’, *Computational Intelligence*, **14**, 498–562, (1998).
- [9] Henry Prakken, ‘A study of accrual of arguments, with applications to evidential reasoning’, in *Proceedings of the tenth International Conference on Artificial Intelligence and Law*, pp. 85–94, (2005).
- [10] Henry Prakken and Giovanni Sartor, ‘A dialectical model of assessing conflicting arguments in legal reasoning’, *Artificial Intelligence and Law*, **4**, 331–368, (1996).
- [11] Henry Prakken and Giovanni Sartor, *Computational Logic: Logic Programming and Beyond. Essays In Honour of Robert A. Kowalski, Part II*, volume 2048 of *Lecture Notes in Computer Science*, 342–380, Springer-Verlag, Berlin, 2002.
- [12] Henry Prakken and Gerard Vreeswijk, ‘Logics for defeasible argumentation’, in *Handbook of philosophical logic, 2nd Edition*, eds., D. Gabbay and F. Guenther, volume 4, 218–319, Kluwer Academic Publishers, (2002).
- [13] Guillermo R. Simari and Ronald P. Loui, ‘A mathematical treatment of defeasible reasoning and its implementation’, *Artif. Intell.*, **53**(2–3), 125–157, (1992).
- [14] Katia Sycara, ‘Persuasive argumentation in negotiation’, *Theory and Decision*, **28**(3), 203–242, (May 1990).
- [15] Bart Verheij, ‘DefLog: On the logical interpretation of *prima facie* justified assumptions’, *Journal of Logic and Computation*, **13**(3), 319–416, (2003).
- [16] Douglas N. Walton, *Legal argumentation and evidence*, Penn State Press, 2002.
- [17] Douglas N. Walton and Erik C. W. Krabbe, *Commitment in Dialogue*, State University of New York Press, 1995.

Auction Mechanisms for Efficient Advertisement Selection on Public Displays

Terry Payne and Ester David and Nicholas R. Jennings and Matthew Sharifi¹

Abstract. Public electronic displays can be used as an advertising medium when space is a scarce resource, and it is desirable to expose many adverts to as wide an audience as possible. Although the efficiency of such advertising systems can be improved if the display is aware of the identity and interests of the audience, this knowledge is difficult to acquire when users are not actively interacting with the display. To this end, we present *BluScreen*, an intelligent public display, which selects and displays adverts in response to users detected in the audience. Here, users are identified and their advert viewing history tracked, by detecting any Bluetooth-enabled devices they are carrying (e.g. phones, PDAs, etc.). Within *BluScreen* we have implemented an agent system that utilises an auction-based marketplace to efficiently select adverts for the display, and deployed this within an installation in our Department. We demonstrate, by means of an empirical evaluation, that the performance of this auction-based mechanism when used with our proposed bidding strategy, efficiently selects the best adverts in response to the audience presence. We benchmarked our advertising method with two other commonly applied selection methods for displaying adverts on public displays; specifically the *Round-Robin* and the *Random* approaches. The results show that our auction-based approach, that utilised the novel use of Bluetooth detection, outperforms these two methods by up to 64%.

1 Introduction

Public electronic displays² are increasingly being used to provide information to users, to entertain (e.g. showing news bulletins), or to advertise products within public environments such as airports, city centres, and retail stores. Within these displays, advertisers typically utilise a variety of delivery methods to maximise the number of different adverts displayed, and thus increase their overall exposure to target audiences [8]. However, these methods are typically naïve and do not take into account the current audience.

On the other hand, a number of *interactive* public displays have been proposed that support communication with a user through active use of handheld devices such as PDAs or phones, or to a closed set of known users with pre-defined interests and requirements [3, 7]. Such systems assume prior knowledge about the target audience, and require either that a single user has exclusive access to the display, or that users carry specific *tracking* devices [4, 10] so that their presence can be identified. These approaches fail to work in public spaces, where no prior knowledge exists regarding users who may view the display, and where such displays need to react to the presence of several users simultaneously.

In contrast, we have developed an intelligent public display that utilises a novel approach to detect nearby users, in order to improve the selection of adverts for display. The goal of the selection is to maximise the exposure of as many adverts as possible to as wide an audience as possible (i.e. to maximise the number of distinct adverts seen by the population of users). In doing so, the main advantage of our system design is that it achieves this goal without: (i) any prior knowledge on the audience, (ii) the need for any specific action by the user, or (iii) the need for any client-based software. Moreover, unlike interactive public displays, our detection technology facilitates an awareness of several devices simultaneously.

As no direct feedback is received from the audience and the only knowledge available is based on the past observations of user presence, one of the key challenges of our system is to predict which advert is likely to gain the highest exposure during the next advertising cycle. To approximate this prediction, our system utilizes history information of past users' exposure to certain sets of adverts (so that we don't repeat material they have already seen), along with the information about what users are currently viewing on the display. In particular, we developed a multi-agent auction-based mechanism to efficiently select an advert for each advertising time slot. In this system, each agent represents a stakeholder that wishes to advertise, and it is provided with a bidding strategy that utilises a heuristic to predict future advert exposure, based on the expected audience composition.

In order to evaluate our design, we deployed a prototype of the system in our department (where it advertises information about research projects, press releases and announcements), and developed a simulator which models user behaviour. Here we report empirical results that show that the auction is more efficient in selecting adverts to maximise exposure of this advertising material to a set of users in the shortest time possible. Specifically, the auction requires, on average, 36% fewer advertising cycles to display all the adverts to each user, when compared to the *Round-Robin* approach (or 64% fewer adverts when compared to the *Random* selection approach).

In more detail, this paper advances the state of the art by:

1. Deploying the *BluScreen* prototype which synergistically combines a public screen with the novel detection of nearby handheld devices using Bluetooth wireless technology.
2. Developing a multi-agent auction-based marketplace for effectively marketing adverts on the *BluScreen* prototype.
3. Devising a novel heuristic-based bidding strategy that can be used by the agents in the auction mechanism.
4. Benchmarking our method against two commonly used selection strategies, *Round-Robin* and *Random*, to demonstrate that our method can efficiently display the best set of adverts given the current audience.

¹ School of Electronics and Computer Science, University of Southampton, UK, email:{trp, ed, nrj, mns203}@ecs.soton.ac.uk

² An electronic display can change the advert presented over time.

The remainder of this paper is organised as follows: Section 2 discusses related work and motivates the use of agents and auction-based marketplaces within *BluScreen*. The deployed system and use of Bluetooth-devices is discussed in Section 3 and the underlying architecture and auction mechanism are described in Sections 4 and 5. In Section 6, we describe the device simulation and our experimental testbed, and present empirical results. Section 7 concludes.

2 Related Work

Targeted advertising has become prevalent through personal advertising systems, such as recommendation systems or web-based banner adverts [1]. These select content based upon prior knowledge of the individual viewing the material, and such systems work well on personal devices (where the owner's preferences and interests can be gathered and cached locally) or within interactive environments which utilise some form of credential to identify the user (e.g. e-commerce sites such as Amazon.com). Such approaches work well when advertising to an individual user, where a rich user profile exists a priori; in contrast, *BluScreen* selects adverts based on the presence of several users, where no profile data is available.

CASy [2] extends this targeted advertising metaphor by using a Vickrey auction mechanism to sell advertising space within a *modelled* electronic shopping mall. The auction was used to rank a set of possible advertisements provided by different retail outlets, and select the top ranking advertisements for presentation on public displays. Feedback is provided through subsequent sales information, allowing the model to build up a profile of a user's preferences. Although a number of different consumer models were evaluated, both with static and adaptive bidding strategies, the system has not been deployed, and is not suitable for advertising to many individuals simultaneously, as it requires explicit interaction with a single user to initially acquire the user's interest and preferences.

The Hermes Photo Display [3] is an example of a community-based display that interacts with users via Bluetooth-enabled phones. Although not used for advertising, users can share photos with each other by either uploading them from a phone to the display, or downloading shared photos to their phone. No specific client software is required on the phone, as the system utilises the fact that many modern mobile phones can share multimedia via Bluetooth. Although this display can be used to share photos; unlike *BluScreen*, the presented content is static, and requires direct user interaction (i.e. via a touch-screen) to browse the photos.

Groupcast [7] was a project that responded to the local audience within a corporate environment to display bespoke media. It had the advantage of knowing a priori the profiles of several members of the audience, and thus could exploit this pre-defined knowledge. User identification was based on an infrared badge system and embedded sensors within an office environment. When several users passed by the display, Groupcast compared the user's profiles to identify common areas of interest. Whereas *BluScreen* selects the best content to maximise exposure to the current audience, Groupcast would try to present content that matched this common interest. However, acquiring such content a priori, and determining high fidelity user profiles were found to be major stumbling blocks, and the mutual content selection was finally abandoned.

3 The BluScreen Prototype

The notion that user presence in front of the intelligent display can be identified remotely, has been explored using several methods within

different pervasive projects. However, most require the deployment of specialised hardware, such as *Active Badges*, [4, 10]. The Bluetooth wireless protocol, characterised by its relative maturity, market penetration³, and emphasis on short-range communication (e.g. 2-3 meters), has emerged as an alternative, generic mechanism for identifying or interacting with users through small, handheld devices.

BluScreen utilises Bluetooth-enabled devices as proxies for identifying users. Bluetooth sensors attached to an intelligent screen can be used to identify these devices⁴ in the local environment, and can record the encounters as a collocation event in terms of location and duration. The duration of this collocation event is assumed to relate to a possible level of interest in the displayed material; e.g. a user who is interested in the current advertising material will linger at the display during the advert.

A *BluScreen* prototype has been developed and deployed to evaluate the feasibility of the auction-based approach. A 23 inch flat-screen display and Bluetooth sensor were deployed outside an office adjacent to the corner of two corridors and an exit (thus maximising visibility to individuals moving within both corridors). The environment was scanned for Bluetooth devices every 20 seconds⁵, and these scans were logged over six months to determine whether or not Bluetooth devices could be used as a proxy for human presence. Twelve adverts were generated, describing a range of topics including research projects, upcoming events, and general information.

Device Type	Unique Samples	Devices
<i>Occasional</i>	< 10	135
<i>Frequent</i>	10-1000	70
<i>Persistent</i>	> 1000	6

Table 1. Number of Bluetooth devices observed at different frequencies over a six month sample period.

Table 1 summarises the number of devices detected, split into three clusters: *occasional*, whereby devices are observed for only a short time (such as visitors, etc.); *frequent* representing devices that regularly pass by the display; and *persistent* which represents devices that are regularly found in proximity to the sensor (such as laptops, etc.) that do not represent individuals passing the screen. Of the 212 unique Bluetooth devices detected, approximately 70 were detected with some degree of regularity, suggesting that Bluetooth detection could be used as a proxy for individuals passing in front of a screen.

4 The Agent Architecture

Several types of agents have been designed within the *BluScreen* architecture that we developed (illustrated in Figure 1):

Bluetooth Device-Detection Agent This agent monitors the environment by periodically sampling it to determine if there are any Bluetooth devices detectable. Current samples are then compared to previous samples to determine if new devices are present or existing devices have left the environment, and to update the exposure rate of existing devices.

Advertising Agent Each Advertising Agent represents a single *Advertisement* and is responsible for effectively purchasing advertising space by generating bids based on their predicted valuation of the upcoming advertising cycle. This predicted valuation may be

³ In October, 2005, ElectronicsWeekly.com reported that "...Bluetooth is expected to ship around 270 million units during 2005..."

⁴ Only Bluetooth devices in discovery mode are detectable.

⁵ The choice of a 20 second scanning cycle was determined by evaluating different scanning cycle lengths with varying numbers of nearby devices.

based on the expected exposure to an arbitrary audience (i.e. being seen by as many users as possible) or could be more selective (e.g. by identifying whether or not users have a preference for the type of material contained in the agent's advertisement). However, our deployed prototype described here currently implements the former method. Thus, each advertising agent maintains a history of successful advertising cycles (i.e. for which cycles an agent won an auction, and thus displayed its advert on the *BluScreen* display), and also the time (and duration) of each Bluetooth-device that was detected by that display during each advertising cycle.

Marketplace Agent The marketplace agent facilitates the sale of advertising time slots. A repetitive second-price sealed-bid auction [9] is used as a selection mechanism (this choice is justified below) to determine which advertising agent (if any) will display its advert on the next time slot, and its corresponding payment.

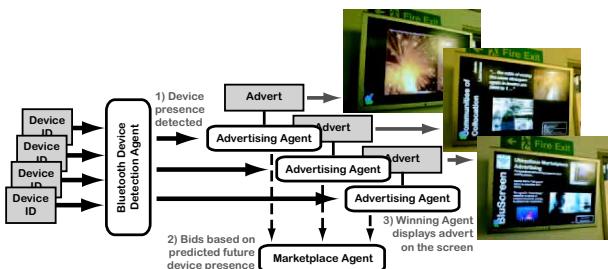


Figure 1. The *BluScreen* Agent Architecture for a single intelligent display

5 Auction Mechanism

BluScreen is designed to support a scaleable and a dynamic advertising framework, while maximising the exposure of as many adverts as possible to as wide an audience as possible, within a knowledge-poor environment. The main principle of our design is to distribute the control of the content displayed, in a way that no single entity can dictate who will advertise next. In contrast, the system, as a whole, will decide who will be the most profitable agent (i.e., expected to gain the highest exposure by displaying its advert in the next advertising cycle) and therefore will be awarded the facility of advertising in that cycle. Note that although the advertising agents are self interested (i.e. aim to maximise their own exposure to as large an audience as possible using their fixed budget), this does not contradict our desired overall design goal, since in our context, we assume that each agent will bid a value that reflects its expected exposure from displaying on the next cycle. Specifically, we have implemented a repetitive, second-price sealed-bid auction that takes place before each of the advertising cycles⁶. The winner of each auction is the advertising agent who placed the highest bid, and it is charged for the second highest bid. As truth-revealing has been shown to be a

⁶ We chose this mechanism for several reasons. As one of the critical requirements of this system is to operate in a timely manner, only *one-shot* auctions were considered. Now, *first-price* and *second-price* sealed-bid auctions are the most popular forms of such auctions. Here we chose the second-price for its superiority in terms of its economically desired properties. In particular, it has a dominant strategy of truth revealing. Such a property is highly desired as it frees the agents from requiring expensive and costly strategies in terms of time and computation. In our context, this is even more significant as the system should react in a timely manner. Moreover, the second-price sealed-bid auction is efficient in the sense that the auctioned item, in our case the right to advertise on the next advertising cycle, is awarded to the bidder that values it the most. Our choice can also be supported by the fact that the continuously repeated second-price sealed-bid auction is also being used for web advertising using search engines like Google and Overture [5].

dominant strategy in this case, the effective local decisions of each individual agent contribute towards effective overall system.

In this section we start by describing the *BluScreen* model and the mechanism we developed, including the strategies of the advertising agents given their valuation for the next advertising cycle's auction. However, as our system is a knowledge-poor environment where no direct feedback is collected from the users (instead only a limited history and the current status of the detected devices is available), in the final part of this section we provide the agent with a heuristic method to generate this value.

5.1 The Auction Model

In our model we assume that there is a single *BluScreen* instance available for advertising which is managed by the marketplace agent⁷. The advertising time is divided into discrete intervals which we term the *advertising cycle*, C^i . During each such cycle, only one agent can display its corresponding advert, and during this time only this agent has the ability to collect information about devices which were detected by the *BluScreen*. However, at the end of each advertising cycle, the marketplace agent informs all the advertising agents about any devices detected in front of the screen at that time.

Each advertising agent, a_j , is assumed to be self interested, and to be managed by a different stake-holder which credits it with a fixed budget for advertising. For each advertising cycle, each advertising agent generates its private value (discussed in detail below) and places a corresponding bid in terms of some currency. Once an agent has fully spent its budget, it is deleted from the system. On the other hand, at any point in time, new agents may be added to the system. The goal of each advertising agent is to utilise its budget in a way that maximises its exposure.

The marketplace agent acts as the auctioneer and runs an auction before each of the advertising cycles. As part of the auctioneer duties, it sets a reservation price that represents the minimum acceptable bid⁸. In the case that none of the agents place bids that meet this price, a default advert is displayed. The marketplace agent is also assumed to be trusted by the advertising agents. This feature is necessary for the proper behaviour of the system, as in a second-price sealed-bid auction no one observes the whole set of bids except the auctioneer.

5.2 Bidding Strategy for the Advertising Agents

In our model, agents participate in a *repetitive*, second-price sealed bid auction whilst facing budget constraints. For non-repetitive variants of this auction, it has been shown that the bidding strategy:

$$\beta(a_j) = \min \{ \text{budget}(a_j), v \} \quad (1)$$

is a **dominant strategy** [6], where $\text{budget}(a_j)$ is the current balance of agent a_j and v is its private valuation. Now, it is easy to see that this holds for the repetitive case as well, as in each round of the second-price sealed-bid auction, a new advertising cycle is auctioned. However, there exists a dependence with previous rounds in terms of the devices that were exposed in the past, and those that are currently in front of the screen. However, this dependence is concealed in the valuation of the bidder for the next advertising cycle.

⁷ This assumption is made to simplify our task of validating the correctness and the efficiency of the proposed mechanism and strategy. This assumption will be relaxed in future work.

⁸ The reservation price is used to avoid selling the advertising space for free, and to ensure that when competition is low, there is a minimum income.

But, once the agent has generated its valuation for the next advertising cycle, its dominant strategy is truth telling.

Although the agents utilise truth telling as a dominant strategy, they still face a very challenging problem of being able to estimate their *expected exposure*, as they are provided only with information about past and current audience composition (in terms of detectable Bluetooth devices), and from that alone they need to estimate their future exposure. In the next subsection we describe a heuristic method we developed for the bidder valuation estimation task.

5.3 Heuristics for Estimating Private Values

Here we provide an advertising agent, a_j , with a heuristic strategy for generating its valuation (expected utility) for the next advertising cycle, C^{i+1} . Recall that the main target of an agent a_j is to maximise its exposure to as large an audience as possible, given its financial constraints. The core principle of this heuristic is to utilise the information an agent has about past devices' exposure, to predict the future expected exposure. Specifically, each time an advertising agent a_j has to make a decision about its valuation for the next cycle C^{i+1} , it has two types of information on which to base its decision:

- (i) history observation, $H(a_j)$, of exposed devices which were collected during the advertising cycles it won, $WonCycles(a_j)$, in the past, $H(a_j) = \{(C^t, d, x)\}$ where $C^t \in WonCycles(a_j)$, d is the device id, and x is the exposure duration; and
- (ii) the current set of detected devices which were in front of the screen at the end of C^i , termed $end(C^i)$.

Using this information, an advertising agent, a_j , assumes that the devices that will be in front of the screen in C^{i+1} are $end(C^i)$. Therefore, we propose that a_j will search through its history to check if these devices were exposed to its advert in the past, and, if so, for how long⁹. If a device was exposed to the same advert during several different advertising cycles, then we propose to consider only the one in which it was exposed to the most. Formally, a_j 's valuation for C^{i+1} is:

$$v(a_j, C^{i+1}) = \sum_{d \in end(C^i)} 1 - \max \{x \mid \{C^t, d, x\} \in H(a_j)\}^{10} \quad (2)$$

Now, we believe this is the best heuristic an agent can use, given its limited information it faces currently. We intend to incorporate more sophisticated mechanisms for generating statistics within future versions of *BluScreen*, that can provide the agents with more relevant information, such as the rate of detected devices as a function of the time in the day. Given that information, agents will be able to apply learning methods to improve their predictions.

6 Empirical Evaluation

To evaluate the bidding strategy described in the previous Section within a controlled environment, a simulation was developed. This is based on the same architecture as the deployed *BluScreen* instance,

⁹ If a device only observes part of an advert, a_j still has an incentive to expose its advert to this device, as we make the assumption that the advert may be dynamic (e.g. contain video). Therefore a device is only exposed to all the information contained in the advert if it observes the advert for the whole advertising cycle. However, having observed a partial advert will diminish the devices' contribution towards bids in future cycles.

¹⁰ In the case where a device is not exposed to any of the previous displays, we assume the default value of x is zero.

but it additionally supports the modelling of users in the audience, in terms of Bluetooth devices present¹¹.

The *BluScreen* simulation modelled devices' behaviour (i.e. as a proxy for a user) in terms of their likelihood of arriving at, and subsequently remaining at, a *BluScreen* display given the currently displayed advertisement. The *device model* was defined with the following assumptions:

- Device presence is measured in discrete sample-intervals, to simulate observations made by the Bluetooth sensor;
- The duration of an advert is assumed to be equal to a whole number of sample-intervals;
- An advert is considered as *fully seen* only when a device has been present for the whole duration of the advert;
- Devices can arrive at any point during an advertising cycle, whereby the probability that a device will arrive is P_{arrive} ;
- The probability a device may leave without observing the subsequent advert is P_{depart} . A device will automatically leave if it has *fully seen* the subsequent advert;
- Both P_{depart} and P_{arrive} assume a uniform distribution.

Two alternate selection methods were compared with the auction: **Round-Robin** selection and **Random** selection. The former is a familiar mechanism used to repeatedly cycle through a number of adverts (in order). Given our device model, this approach represents the optimal selection mechanism when $P_{depart} = 0$ and $P_{arrive} = 1$. The latter mechanism randomly selects a new advert to display at the beginning of each new advertising cycle, independently of what has been previously selected. This method was selected as a baseline against which the other worst-case methods can be compared.

To simplify this analysis, each experiment consisted of ten adverts of equal duration (i.e. six samples which is equivalent to a 2 minute advert), represented by ten advertising agents. In each experiment, adverts are selected and presented until every device has *fully seen* all ten adverts. To ensure statistical stability, the results of each experiment are averaged over 10,000 test runs, and the mean results are reported. Where necessary, a Student's t-test is used to confirm statistical significance at the 95% confidence level.

6.1 Experimental Results

In our first experiment, we examine the behaviour of each selection mechanism as the number of devices present increases (Figure 2). The number of devices, N_d , was varied between 1 and 100, and the device behaviour was defined using $P_{depart} = 0.05$ and $P_{arrive} = 0.5$. The graph supports the hypothesis that advert selection using the *BluScreen auction* is statistically more significant than either *Round-Robin* or *Random* (assuming the modelling parameters defined above). Specifically, as N_d increases, there is a corresponding exponential rise in the number of required adverts. The mean number of adverts required by the *BluScreen auction* is lower than the *Round-Robin* selection mechanism or the *Random* selection mechanism for all numbers of devices tested; e.g. for $N_d = 50$, the auction method required a mean of 40.24 ± 0.06 advertising cycles to display all 10 adverts to all the devices, compared to *Round-Robin* (64.16 ± 0.18) or *Random* (108.50 ± 0.28). This suggests that although each selection method can scale, a single device will be exposed to all the adverts in typically 36% fewer advertising cycles than

¹¹ Although the deployed prototype could be used to test the auction mechanism based on observed real-world events, factors such as the location of the screen, variances in the advertised material, and the number of detectable devices can affect the resulting performance.

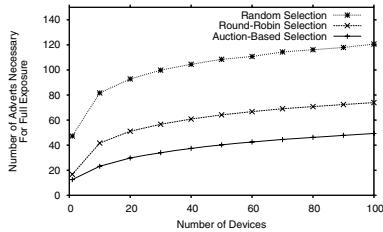


Figure 2. The effect of varying the number of observed devices; $P_{depart} = 0.05$, $P_{arrive} = 0.5$.

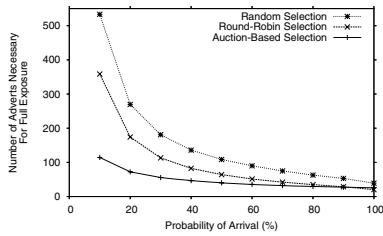


Figure 3. The effect of varying the probability of a device arriving during an advertising cycle. $P_{depart} = 0.05$, $N_d = 50$.

the *Round-Robin* approach, or 64% fewer advertising cycles than the *Random* approach when using the *BluScreen auction*.

The second two experiments therefore explore how the efficiency of the different selection mechanisms change as the device model is varied. In each of these experiments, $N_d = 50$. In particular, Figure 3 plots the results for $0 < P_{arrival} \leq 1$. These results show that for low values of $P_{arrival}$, the number of adverts required is significantly lower for *BluScreen auction* than for either of the other methods considered. However, as the probability increases towards 100%, the efficiency of both *BluScreen auction* and *Round-Robin* converge. This result is expected, as when all devices are guaranteed to be present, *BluScreen auction* will select each advert in turn (depending on the tie-breaking strategy (e.g. random) used by the auctioneer), and thus exhibit the same behaviour as *Round-Robin*.

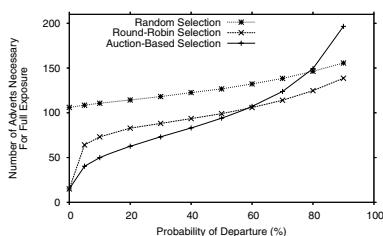


Figure 4. The effect of varying the probability of a device leaving the audience during an advertising cycle. $P_{arrive} = 0.5$, $N_d = 50$.

One artefact of assuming low departure probabilities is that once a device has arrived, it is unlikely to depart, and hence is more likely to remain through a complete cycle of adverts (in the case of *Round-Robin*), or its presence be accurately predicted (when using *BluScreen auction*). Therefore, as P_{depart} increases, the efficiency of both approaches should degrade. Figure 4 illustrates this. As one would expect, the performance of both *BluScreen auction* and *Round-Robin* are statistically equivalent when $P_{depart} = 0$. However, the number of adverts required increases sharply in the range $P_{depart} = [0.01 \dots 0.1]$; and then rises almost linearly until $P_{depart} = 0.5$. Beyond this value, the rise becomes logarithmic. This rise is more significant for the auction approach, and may be due to poor estimates in future device presence, leading to erroneous pri-

vate value predictions. In contrast, the rise in the number of adverts required for *Random* is almost linear; this result is not surprising as the selection of each advert is independent of the audience composition, and hence is unaffected by the departure of a device.

7 Conclusions

In this paper, we addressed the problem of unobtrusively providing audience composition data to an intelligent public display, to select the optimal advert that maximises exposure of a set of adverts. To this end, we proposed the novel use of Bluetooth wireless technology to detect nearby handheld devices, and designed an auction-based marketplace and bidding heuristic which predicted the level of expected exposure of an advert given the detected audience and knowledge of previous exposure the audience members had to different adverts. A prototype public display that utilises this marketplace was implemented and deployed in our Department. The logs generated by this system indicated that a large number of Bluetooth-enabled devices could be detected, and that a significant subset was detected on a regular basis. Moreover, we implemented a simulation environment to facilitate controlled evaluation of the auction mechanism, when compared with other commonly used advert selection mechanisms, and showed that in most cases the auction-based approach outperforms the other commonly used approaches. This approach has particular relevance for advertising in environments where users only linger for short periods, but often return (such as railway platforms etc.).

In future work, we plan to extend our approach to work with multiple displays, to increase the exposure of adverts to a larger community, and to build user preferences based on the duration users spend viewing different adverts that are categorised (thus taking into account inferred interest, as well as exposure).

ACKNOWLEDGEMENTS

This research was partially funded through an internal scholarship provided by the School of Electronics and Computer Science.

REFERENCES

- [1] Ali Amiri and Syam Menon, ‘Efficient scheduling of internet banner advertisements’, *ACM Trans. on Internet Tech.*, **3**(4), 334–346, (2003).
- [2] Sander M. Bohte, Enrico Gerdin, and Han La Poutre, ‘Market-based recommendation: Agents that compete for consumer attention’, *ACM Trans. on Internet Tech.*, **4**(4), 420–448, (2004).
- [3] Keith Cheverst, Alan Dix, Daniel Fitton, Chris Kray, Mark Rouncefield, Corina Sas, George Sasilis-Lagoudakis, and Jennifer G. Sheridan, ‘Exploring bluetooth based mobile phone interaction with the hermes photo display’, in *MobileHCI ’05: Proc. 7th Int. Conf. on Human computer interaction with mobile devices & services*, pp. 47–54, (2005).
- [4] Jeffrey Hightower and Gaetano Borriella, ‘Location systems for ubiquitous computing’, *IEEE Computer*, **34**(8), 57–66, (2001).
- [5] Brendan Kitts and Benjamin Leblanc, ‘Optimal bidding on keyword auctions’, *Electronic Market*, **14**(3), 186–201, (2004).
- [6] Vijay Krishna, *Auction Theory*, Academic Press, 2002.
- [7] Joseph F. McCarthy, Tony J. Costa, and Edy S. Liongsoari, ‘Unicast, outcast & groupcast: Three steps toward ubiquitous, peripheral displays’, in *UbiComp ’01: Proc. 3rd Int. Conf. on Ubiquitous Computing*, pp. 332–345, London, UK, (2001). Springer-Verlag.
- [8] Anand Ranganathan and Roy H. Campbell, ‘Advertising in a pervasive computing environment’, in *WMC ’02: Proc. 2nd Int. Wkshp. on Mobile commerce*, pp. 10–14, New York, NY, USA, (2002). ACM Press.
- [9] W. Vickrey, ‘Counterspeculation, Auctions and Competitive Sealed Tenders’, *Journal of Finance*, 8–37, (1961).
- [10] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons, ‘The active badge location system’, *ACM Trans. on Information Systems*, **10**(1), 91–102, (1992).

Verifying Interlevel Relations within Multi-Agent Systems

Alexei Sharpanskykh and Jan Treur¹

Abstract. An approach to handle the complex dynamics of a multi-agent system is based on distinguishing aggregation levels by structuring the system into parts or components. The behavior of every aggregation level is specified by a set of dynamic properties for components and interactions at that level, expressed in some (temporal) language. The dynamic properties of higher aggregation levels in principle can be logically related to dynamic properties of lower levels. This asks for identification and verification of such interlevel relations. In this article it is shown how this problem can be addressed using model checking techniques.

1 INTRODUCTION

Often dynamics of a multi-agent system is described by a behavioral specification, which consists of dynamic properties of elements of the multi-agent system (i.e., agents, interaction relations, and an environment). Usually, these properties are expressed as formulae in some (temporal) language. Even if the behavioral description of a single element is simple, the general dynamics of the whole multi-agent system is often difficult to analyze. In particular, with increase of the number of elements within the multi-agent system, the complexity of the dynamics of the system grows considerably. In order to analyze the behavior of a complex multi-agent system (e.g., for critical domains such as air traffic control and health care), appropriate approaches for handling the dynamics of the multi-agent system are important.

One of the approaches to manage complex dynamics is by distinguishing different *aggregation levels*, based on a clustering of a multi-agent system into parts or components with further specification of their dynamics and relations between them; e.g., [4]. At the lowest aggregation level a component is an agent or an environmental object (e.g., a database), with which agents interact. Further, at higher aggregation levels a component has the form of either a group of agents or a multi-agent system as a whole. In the simplest case two levels can be distinguished: the lower level at which agents interact and the higher level, where the whole multi-agent system is considered as one component. In the general case the number of aggregation levels is not restricted. At every aggregation level the behavior of a component is described by a set of dynamic properties. The dynamic properties of a component of a higher aggregation level can be logically related by an *interlevel relation* to dynamic properties of components of an adjacent lower

aggregation level. This interlevel relation takes the form that a number of properties of the lower level logically entail the properties of the higher level component.

Identifying interlevel relations is usually achieved by applying informal or semi-formal early requirements engineering techniques; e.g., *i** [8] and SADT [7]. To formally prove that the identified interlevel relations are indeed correct, model checking techniques [1, 9] may be of use. The idea is that the lower level properties in an interlevel relation are used as a system specification, whereas the higher level properties are checked for this system specification. However, model checking techniques are only suitable for systems specified as finite-state concurrent systems. In the general case, at any aggregation level a behavioral specification for a multi-agent system component consists of dynamic properties expressed by possibly complex temporal relations, which do not allow direct application of automatic model checking procedures. In order to apply model checking techniques it is needed to transform an original behavioral specification of the lower aggregation level into a model based on a finite state transition system. In order to obtain this, as a first step a behavioral description for the lower aggregation level is replaced by one in executable temporal format. After that, using an automated procedure an executable temporal specification is translated into a general finite state transition system format that consists of standard transition rules. Such a representation can be easily translated into an input format of one of the existing model checkers. Then, using model checking techniques it is possible to prove (or refute) automatically that the interlevel relations between dynamic properties of adjacent aggregation levels expressed as specification in some temporal language hold.

The proposed approach has similarities with compositional reasoning and verification techniques [3, 4] in the way how it handles complex dynamics of a system. However, it differs from the latter in distinguishing multiple aggregation levels in a system specification and representing verification of system dynamics as a problem of checking logic entailment relations between dynamic properties of adjacent aggregation levels. Other approaches for verifying multi-agent system dynamics are based on modal temporal logics, automatically translated into executable format by techniques from [2], and verified by clausal resolution methods.

In the next section the concepts for formal specification of a multi-agent system's dynamics and the temporal language used are briefly introduced. After that, in Section 3 the transformation procedure from a behavioral specification into a finite state transition system description is described and illustrated by means of an example. The paper ends with a discussion in Section 4.

¹ Vrije Universiteit Amsterdam, Dept of AI, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. E-mail: {sharp, treur}@few.vu.nl URL: <http://www.few.vu.nl/~{sharp,treur}>

2 MODELING OF DYNAMIC PROPERTIES

Components can be active (e.g., an agent) or passive (e.g., database). An active component represents an autonomous entity that interacts with an environment and with other components. Active components interact with each other by communicating messages to each other (at the mental level) and performing actions (at the physical level). An environment can be considered as a set of passive components with certain properties and states. Components interact with each other via *input* and *output* (interface) states. At its input an active component receives observations from an environment or communications from other components whereas at its output it generates communications to other components or actions in an environment.

From the external perspective behavior of a component is characterized by a set of dynamic properties, which represent relations over time between its input and output states. Dynamic properties are specified in the Temporal Trace Language (TTL) [4, 11]. TTL is a variant of order-sorted predicate logic [6] and has some similarities with Situation Calculus [10] and Event Calculus [5]. Whereas the standard multi-sorted predicate logic is a language to reason about static properties only, TTL is an extension of such language with facilities for reasoning about the dynamic properties of arbitrary systems.

The state properties are expressed using a standard multi-sorted first-order predicate language with a signature, which consists of a number of sorts, sorted constants, variables, functions and predicates. Every component A has assigned an interaction state ontology $\text{InteractionOnt}(A)$ for its input and output states. Specifically, using an ontology InteractionOnt one can define observations of state properties, communications, and actions.

For enabling dynamic reasoning TTL includes special sorts: TIME (a set of linearly ordered time points), STATE (a set of all state names of a system), TRACE (a set of all trace names; a trace or a trajectory can be thought of as a timeline with a state for each time point), STATPROP (a set of all state property names), and VALUE (an ordered set of numbers). Furthermore, for every sort S from the state language the following TTL sorts exist: the sort S^{VARS} , which contains all variable names of sort S ; the sort S^{TERMS} , which contains names of all ground terms, constructed using sort S ; sorts S^{GTERMS} and S^{VARS} are subsorts of sort S^{TERMS} .

In TTL, formulae of the state language are used as objects. To provide names of object language formulae ϕ in TTL the operator $(*)$ is used (written as ϕ^*), which maps variable sets, term sets and formula sets of the state language to the elements of TTL sorts S^{GTERMS} , S^{TERMS} , S^{VARS} and STATPROP . The state language and TTL define disjoint sets of expressions. Therefore, in TTL formulae we shall use the same notations for the elements of the object language (i.e., constants, variables, functions, predicates) and for their names in TTL without introducing any ambiguity. Further we shall use t with subscripts and superscripts for variables of the sort TIME ; and γ with subscripts and superscripts for variables of the sort TRACE .

For the explicit indication of an aspect of a state for a component, to which a state property is related, sorts ASPECT_COMPONENT (a set of the component aspects of a system; i.e., input, output, internal); COMPONENT (a set of all component names of a system); $\text{COMPONENT_STATE_ASPECT}$ (a set of all names of aspects of all component states) and a function symbol

$\text{comp_aspect: ASPECT_COMPONENT} \times \text{COMPONENT} \rightarrow \text{COMPONENT_STATE_ASPECT}$

are used.

A state for a component is described by a function symbol state of type $\text{TRACE} \times \text{TIME} \times \text{COMPONENT_STATE_ASPECT} \rightarrow \text{STATE}$.

The set of function symbols of TTL includes $\wedge, \vee, \rightarrow, \leftrightarrow: \text{STATPROP} \times \text{STATPROP} \rightarrow \text{STATPROP}$; not: $\text{STATPROP} \rightarrow \text{STATPROP}$, $\forall, \exists: S^{\text{VARS}} \times \text{STATPROP} \rightarrow \text{STATPROP}$, which counterparts in the state language are boolean propositional connectives and quantifiers. Further we shall use $\wedge, \vee, \rightarrow, \leftrightarrow$ in infix notation and \forall, \exists in prefix notation for better readability.

Notice that also within states statements about time can be made (e.g., in state properties representing memory). To relate time within a state property (sort LTIME) to time external to states (sort TIME) a function $\text{present_time}: \text{LTIME}^{\text{TERMS}} \rightarrow \text{STATPROP}$ is used. Furthermore, for the purposes of this paper it is assumed that $\text{LTIME}^{\text{GTERMS}} = \text{TIME}$ and $\text{LVALUE}^{\text{GTERMS}} = \text{VALUE}$ (LVALUE is a sort of the state language, which is a set of numbers). We shall use u with subscripts and superscripts to denote constants of sort $\text{LTIME}^{\text{VARS}}$. For formalising relations between sorts VALUE and TIME function symbols $-, +, /, \cdot: \text{TIME} \times \text{VALUE} \rightarrow \text{TIME}$ are introduced. And for sorts $\text{LVALUE}^{\text{TERMS}}$ and $\text{LTIME}^{\text{TERMS}}$ the function symbols $-, +, /, \cdot$ are overloaded: $\text{LTIME}^{\text{TERMS}} \times \text{LVALUE}^{\text{TERMS}} \rightarrow \text{STATPROP}$.

The states of a component are related to names of state properties via the formally defined satisfaction relation denoted by the infix predicate \models (or denoted by the prefix predicate holds): $\text{state}(\gamma, t, \text{output}(A)) \models p$ (or $\text{holds}(\text{state}(\gamma, t, \text{output}(A)))$), which denotes that the state property with a name p holds in trace γ at time point t at the output state of component A. Sometimes, when the indication of a component aspect is not necessary, this relation will be used without the third argument: $\text{state}(\gamma, t) \models p$. Both $\text{state}(\gamma, t, \text{output}(A))$ and p are terms of TTL. In general, TTL terms are constructed by induction in a standard way from variables, constants and function symbols typed with all before mentioned TTL sorts.

Transition relations between states are described by dynamic properties, which are expressed by TTL-formulae. The set of *atomic TTL-formulae* is defined as:

- (1) If v_1 is a term of sort STATE , and u_1 is a term of the sort STATPROP , then $\text{holds}(v_1, u_1)$ is an atomic TTL formula.
- (2) If t_1, t_2 are terms of any TTL sort, then $t_1=t_2$ is an atomic TTL formula.
- (3) If t_1, t_2 are terms of sort TIME , then $t_1 < t_2$ is an atomic TTL formula.

The set of *well-formed TTL-formulae* is defined inductively in a standard way using boolean propositional connectives and quantifiers. TTL has semantics of the order-sorted predicate logic. A more detailed specification of the syntax and the semantics for the TTL (including the axiomatic basis) is given in [11].

Dynamic properties to model a behavioral specification are assumed to be specified in the form of a logical implication from a temporal input pattern to a temporal output pattern. The consequent parts of dynamic properties do not contain any disjunctions in order to prevent non-determinism in behavior. It is a necessary assumption for enabling verification of a system using existing model checking techniques and tools. Past, interval and future statements are defined as follows:

- a) A *past statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\phi_p(\gamma, t)$ in TTL, such that each time variable s different from t is restricted to the time interval before t : for every time quantifier for a time variable s a restriction of the form $s \leq t$, or $s < t$ is required within the statement.
- b) A *future statement* for a trace γ and a time point t over state ontology Ont is a temporal statement $\phi_f(\gamma, t)$ in TTL, such that for every quantified time variable s , different from t a restriction of the form $s \geq t$, or $s > t$ is required within the statement.

c) An *interval statement* for a trace γ and time points t_1 and t_2 over state ontology Ont is a temporal statement $\phi(\gamma, t_1, t_2)$ in TTL, that is a past statement for t_2 and a future statement for t_1 .

An executable specification of the dynamics of a component consists of a set of dynamic properties in an executable temporal language, representing temporal relations between a number of postulated internal states. Internal states of a component A are described using a postulated internal state ontology $\text{InternalOnt}(A)$. In cognitive sciences, which have been used as a source of inspiration, it is often assumed that an agent maintains a memory in the form of some internal model of the history. Furthermore, we assume that internal states are formed on the basis of (input) observations (sensory representations) or communications. For this the function symbol $\text{memory} : \text{LTIME}^{\text{TERMS}} \times \text{STATPROP} \rightarrow \text{STATPROP}$ is used. For example, $\text{memory}(t, \text{observed}(a))$ expresses that the component has memory that it observed a state property a at time point t . Before performing an action or communication it is postulated that a component creates an internal preparation state. For example, $\text{preparation_for}(b)$ represents a preparation of a component to perform an action or a communication b . Each dynamic property in the internal behavioral specification is specified in one of the following *executable* forms:

- (1) $\forall t \text{state}(\gamma, t) \models X \Rightarrow \text{state}(\gamma, t+c) \models Y$ (states relation property)
- (2) $\forall t \text{state}(\gamma, t) \models X \Rightarrow \text{state}(\gamma, t+1) \models X$ (consistency property)
- (3) $\forall t \text{state}(\gamma, t) \models X \Rightarrow \text{state}(\gamma, t) \models Y$ (state relation property),

where c is some integer constant, X and Y are (conjunctions of) names of state properties and $X \neq Y$.

3 TRANSFORMATION PROCEDURE

The procedure described in this section achieves the transformation of a behavioral specification for multi-agent system components into the executable format and subsequently into the representation of a finite state transition system. An external behavioral specification of a component is defined as follows.

Definition (External behavioral specification)

An *external behavioral specification* for a multi-agent system component consists of dynamic properties $\phi(\gamma, t)$ expressed in TTL of the form $[\phi_p(\gamma, t) \Rightarrow \phi_f(\gamma, t)]$, where $\phi_p(\gamma, t)$ is a past statement over the interaction ontology and $\phi_f(\gamma, t)$ is a future statement. The future statement is represented in the form of a conditional behavior: $\phi_f(\gamma, t) \Leftrightarrow \forall t_i > t [\phi_{\text{cond}}(\gamma, t, t_i) \Rightarrow \phi_{\text{bh}}(\gamma, t_i)]$, where $\phi_{\text{cond}}(\gamma, t, t_i)$ is an interval statement over the interaction ontology, which describes a condition for some specified action(s) and/or communication(s), and $\phi_{\text{bh}}(\gamma, t_i)$ is a (conjunction of) future statement(s) for t_i over the output ontology of the form $\text{state}(\gamma, t_i+c) \models \text{output}(a)$, for some integer constant c and action or communication a .

When a past formula $\phi_p(\gamma, t)$ is true for γ at time t , a potential to perform one or more action(s) and/or communication(s) exists. This potential is realized at time t_i when the condition formula $\phi_{\text{cond}}(\gamma, t, t_i)$ becomes true, which leads to the action(s) and/or communication(s) being performed at the time point(s) t_i+c indicated in $\phi_{\text{bh}}(\gamma, t_i)$.

The procedure is applied to the multi-agent system considered at any two adjacent aggregation levels. The behavioral specification of the lower aggregation level is constructed from the dynamic properties of the lower level components. By interlevel relations these properties can be logically related to the properties of higher level components. In order to enable automatic verification of such interlevel relations, the non-executable lower level behavioral specification is automatically replaced by an

executable one. Further, for performing model checking, the executable specification is automatically transformed into a finite state transition system description.

Let $\phi(\gamma, t)$ be a non-executable dynamic property from an external behavioral specification for the component A, for which an executable representation should be found.

The Transformation Procedure

- (1) Identify executable temporal properties, which describe transitions from interaction states to memory states.
- (2) Identify executable temporal properties, which describe transitions from memory states to preparation states for output.
- (3) Specify executable properties, which describe the transition from preparation states to the corresponding output states.
- (4) From the executable properties, identified during steps 1-3, construct a part of the specification $\pi(\gamma, t)$, which describes the internal dynamics of component A, corresponding to property $\phi(\gamma, t)$.
- (5) Apply steps 1-4 to all properties in the behavioral specification of lower level components A. In the end add to the executable specification the dynamic properties, which were initially specified in executable form using an ontology, different than $\text{InteractOnt}(A)$.
- (6) Translate the identified during the steps 1-5 executable rules into the transition system representation.

The details of the described procedure are described by means of an example, in which a multi-agent system for co-operative information gathering is considered at two aggregation levels. At the higher level the multi-agent system as a whole is considered. At the lower level four components and their interactions are considered: two information gathering agents A and B, agent C, and environment component E representing the external world. Each of the agents is able to acquire partial information from an external source (component E) by initiated observations. Each agent can be reactive or proactive with respect to the information acquisition process. An agent is proactive if it is able to start information acquisition independently of requests of any other agents, and an agent is reactive if it requires a request from some other agent to perform information acquisition.

Observations of any agent taken separately are insufficient to draw conclusions of a desired type; however, the combined information of both agents is sufficient. Therefore, the agents need to co-operate to be able to draw conclusions. Each agent can be proactive with respect to the conclusion generation, i.e., after receiving both observation results an agent is capable to generate and communicate a conclusion to agent C. Moreover, an agent can be request pro-active to ask information from another agent, and an agent can be pro-active or reactive in provision of (already acquired) information to the other agent.

For the lower-level components of this example multi-agent system, a number of dynamic properties were identified and formalized in TTL. For the purposes of illustration of the proposed transformation procedure the dynamic property that describes an information provision reactivity of the agent B has been chosen. Informally this property expresses that the agent B generates an information chunk (the constant IC of sort $\text{INFORMATION_CHUNK}^{\text{TERMS}}$) for the agent A if the agent B observes the IC at its input from the environment and at some point in the past B received a request for the IC from the agent A. According to the definition of an external behavioral specification the considered property can be represented in the form $[\phi_p(\gamma, t) \Rightarrow \phi_f(\gamma, t)]$, where $\phi_p(\gamma, t)$ is a formula

$\exists t_2 \leq t \text{state}(\gamma, t_2, \text{input}(B)) \models \text{communicated}(\text{request_from_to_for}(A, B, \text{IC}))$ and $\phi_f(\gamma, t)$ is a formula

$\forall t_1 > t [state(\gamma, t_1, input(B)) \models observed(provided_result_from_to(E, B, IC)) \Rightarrow state(\gamma, t_1 + c, output(B)) \models output(communicated(send_from_to(B, A, IC)))]$
with $\varphi_{cond}(\gamma, t, t_1)$ is

$state(\gamma, t_1, input(B)) \models observed(provided_result_from_to(E, B, IC))$
and $\varphi_{bh}(\gamma, t_1)$ is

$state(\gamma, t_1 + c, output(B)) \models output(communicated(send_from_to(B, A, IC)))$,
where t is the present time point with respect to which the formulae are evaluated and c is some natural number.

Step 1. From interaction states to memory states

The formula $\varphi_{mem}(\gamma, t)$ obtained by replacing all occurrences in $\varphi_p(\gamma, t)$ of subformulae of the form $state(\gamma, t') \models p$ by $state(\gamma, t) \models memory(t', p)$ is called the *memory formula for* $\varphi_p(\gamma, t)$. Thus, a memory formula defines a sequence of past events (i.e., a history; e.g., observations of an external world, actions) for the present time point t . According to Lemma 1 (given in [11]) $\varphi_{mem}(\gamma, t)$ is equivalent to some formula $\delta^*(\gamma, t)$ of the form $state(\gamma, t) \models q_{mem}(t)$, where $q_{mem}(t)$ is called the *normalized memory state formula for* $\varphi_{mem}(\gamma, t)$, which uniquely describes the present state at the time point t by a certain history of events. Moreover, q_{mem} is the state formula $\forall u' [present_time(u') \rightarrow q_{mem}(u')]$. For the considered example $q_{mem}(t)$ for $\varphi_{mem}(\gamma, t)$ is specified as:

$$\exists u_2 \leq t \text{ memory}(u_2, \text{communicated(request_from_to_for(A, B, IC)))}$$

Additionally, memory state persistency properties are composed for all memory atoms. Rules that describe creation and persistence of memory atoms are given in the executable theory from observation states to memory states $Th_{o \rightarrow m}$ (a general description of this and the following theories is given in [11]). For the example:

$$\begin{aligned} \forall t' \text{ state}(\gamma, t', input(B)) \models \text{communicated(request_from_to_for(A, B, IC)))} \Rightarrow \\ \text{state}(\gamma, t', internal(B)) \models \\ \text{memory}(t', \text{communicated(request_from_to_for(A, B, IC)))}) \\ \forall t'' \text{ state}(\gamma, t'', internal(B)) \models \\ \text{memory}(t', \text{communicated(request_from_to_for(A, B, IC)))}) \Rightarrow \\ \text{state}(\gamma, t'' + 1, internal(B)) \models \\ \text{memory}(t', \text{communicated(request_from_to_for(A, B, IC)))}) \end{aligned}$$

Step 2. From memory states to preparation states

Obtain $\varphi_{cmem}(\gamma, t, t_1)$ by replacing all occurrences in $\varphi_{cond}(\gamma, t, t_1)$ of $state(\gamma, t') \models p$ by $state(\gamma, t_1) \models memory(t', p)$. The condition memory formula $\varphi_{cmem}(\gamma, t, t_1)$ contains a history of events, between the time point t , when $\varphi_p(\gamma, t)$ is true and the time point t_1 , when the formula $\varphi_{cond}(\gamma, t, t_1)$ becomes true. Again by Lemma 1 $\varphi_{cmem}(\gamma, t, t_1)$ is equivalent to the formula $state(\gamma, t_1) \models q_{cond}(t, t_1)$, where $q_{cond}(t, t_1)$ is called the *normalized condition state formula for* $\varphi_{cmem}(\gamma, t, t_1)$, and $q_{cond}(t)$ is the state formula $\forall u' [present_time(u') \rightarrow q_{cond}(t, u')]$.

For the considered example $q_{cond}(t, t_1)$ for $\varphi_{cmem}(\gamma, t)$ is obtained as: $memory(t_1, \text{observed(provided_result_from_to}(E, B, IC)))$ and $q_{cond}(t): \forall u' [present_time(u') \rightarrow memory(u', \text{observed(provided_result_from_to}(E, B, IC)))]$.

Obtain $\varphi_{prep}(\gamma, t_1)$ by replacing in $\varphi_{bh}(\gamma, t_1)$ any occurrence of $state(\gamma, t_1 + c) \models output(a)$ by $state(\gamma, t_1) \models preparation_for(output(t_1 + c, a))$, for some number c and action or communication a . The preparation state is created at the same time point t_1 , when the condition for an output $\varphi_{cond}(\gamma, t, t_1)$ is true. By Lemma 1 $\varphi_{prep}(\gamma, t_1)$ is equivalent to the state formula $state(\gamma, t_1) \models q_{prep}(t_1)$, where $q_{prep}(t_1)$ is called the *normalized preparation state formula for* $\varphi_{cond}(\gamma, t_1)$. Moreover, q_{prep} is the state formula $\forall u' [present_time(u')] \rightarrow q_{prep}(u')]$. For the considered example $q_{prep}(t_1)$ is composed as $preparation_for(output(t_1 + c, \text{communicated(send_from_to}(B, A, IC))))$.

Rules, which describe generation and persistence of condition memory states, a transition from the condition to the preparation state, and the preparation state generation and persistence, are given in the executable theory from memory states to preparation states $Th_{m \rightarrow p}$. For the considered example:

$$\begin{aligned} \forall t' [state(\gamma, t', input(B)) \models observed(provided_result_from_to(E, B, IC)) \Rightarrow \\ state(\gamma, t', internal(B)) \models \\ [memory(t', \text{observed(provided_result_from_to}(E, B, IC))) \wedge \\ stimulus_reaction(observed(provided_result_from_to}(E, B, IC)))]] \\ \forall t'' \text{ state}(\gamma, t'', internal(B)) \models \\ memory(t', \text{observed(provided_result_from_to}(E, B, IC))) \Rightarrow \\ state(\gamma, t'' + 1, internal(B)) \models \\ memory(t', \text{observed(provided_result_from_to}(E, B, IC))) \\ \forall t' \text{ state}(\gamma, t') \models \forall u'' [present_time(u'') \rightarrow \\ \exists u_2 [memory(u_2, \text{communicated(request_from_to_for}(A, B, IC)))]] \Rightarrow \\ state(\gamma, t') \models \forall u'' [present_time(u'') \rightarrow [\forall u_1 > u'' \\ [memory(u_1, \text{observed(provided_result_from_to}(E, B, IC))) \rightarrow \\ preparation_for(output(u_1 + c, \text{communicated(send_from_to}(B, A, IC))))]]] \\ \forall t', t \text{ state}(\gamma, t') \models \forall u'' [present_time(u'') \rightarrow [\forall u_1 > u'' \\ [memory(u_1, \text{observed(provided_result_from_to}(E, B, IC))) \rightarrow \\ preparation_for(output(u_1 + c, \text{communicated(send_from_to}(B, A, IC))))] \wedge \\ \wedge \forall u'' [present_time(u'') \rightarrow \\ memory(u'', \text{observed(provided_result_from_to}(E, B, IC)))] \wedge \\ stimulus_reaction(observed(provided_result_from_to}(E, B, IC)))]] \Rightarrow \\ state(\gamma, t', internal(B)) \models \forall u_1 [present_time(u_1) \rightarrow \\ preparation_for(output(u_1 + c, \text{communicated(send_from_to}(B, A, IC))))]] \\ \forall t' \text{ state}(\gamma, t') \models \\ [stimulus_reaction(observed(provided_result_from_to}(E, B, IC))) \wedge \\ not(preparation_for(output(t' + c, \text{communicated(send_from_to}(B, A, IC)))))] \Rightarrow \\ state(\gamma, t' + 1) \models \\ stimulus_reaction(observed(provided_result_from_to}(E, B, IC))) \\ \forall t' \text{ state}(\gamma, t', internal(B)) \models \\ [preparation_for(output(t' + c, \text{communicated(send_from_to}(B, A, IC)))) \wedge \\ not(output(\text{communicated(send_from_to}(B, A, IC))))] \Rightarrow \\ state(\gamma, t' + 1, internal(B)) \models \\ preparation_for(output(t' + c, \text{communicated(send_from_to}(B, A, IC)))) \end{aligned}$$

The auxiliary functions $stimulus_reaction(a)$ are used for reactivation of agent preparation states for generating recurring actions or communications.

Step 3. From preparation states to output states

The preparation state $preparation_for(output(t' + c, a))$ is followed by the output state, created at the time point $t_1 + c$. Rules that describe a transition from preparation to output state(s) are given in the executable theory from the preparation to the output state(s) $Th_{p \rightarrow o}$. For the considered example the following rule holds:

$$\begin{aligned} \forall t' \text{ state}(\gamma, t', internal(B)) \models \\ preparation_for(output(t' + c, \text{communicated(send_from_to}(B, A, IC)))) \Rightarrow \\ state(\gamma, t' + c, output(B)) \models output(\text{communicated(send_from_to}(B, A, IC))) \end{aligned}$$

Step 4. Constructing an executable specification

An executable specification $\pi(\gamma, t)$ for the component A is defined by a union of the dynamic properties from the executable theories $Th_{o \rightarrow m}$, $Th_{m \rightarrow p}$ and $Th_{p \rightarrow o}$, identified during the steps 1-3. For the purposes of analysis of component dynamics the non-executable external behavioral specification is replaced by the executable behavioral specification. The justification of such substitution is based on the theorem in [11].

At Step 5 an executable specification is constructed for the whole external behavioral specification of an agent.

Step 6. Translation of an executable specification into a description of a transition system

For the purpose of practical verification a behavioral specification based on executable temporal logical properties generated at Step 5 is translated into a finite state transition system model (a general procedure is described in [11]).

To translate the executable specification of agent behavior into the transition system representation the atom *present_time* is used. This atom is evaluated to *true* only in a state for the current time point. The executable properties from the executable specification, translated into the transition rules for the considered example are given below:

```

present_time(t) ∧ communicated(request_from_to_for(A,B,IC)) →
    memory(t,communicated(request_from_to_for(A,B,IC)))
present_time(t) ∧ observed(provided_result_from_to(E,B,IC)) →
    memory(t,observed(provided_result_from_to(E,B,IC)) ∧
        stimulus_reaction(observed(provided_result_from_to(E,B,IC))))
memory(t,communicated(request_from_to_for(A,B,IC))) →
    memory(t,communicated(request_from_to_for(A,B,IC)))
memory(t,observed(provided_result_from_to(E,B,IC)) →
    memory(t,observed(provided_result_from_to(E,B,IC)))
present_time(t) ∧
    ∃u2≤t memory(u2,communicated(request_from_to_for(A, B,IC))) →
        conditional_preparation_for(output(communicated(send_from_to(B,A,IC))))
present_time(t) ∧
    conditional_preparation_for(output(communicated(send_from_to(B,A,IC)))) ∧
    memory(t,observed(provided_result_from_to(E,B,IC))) ∧
    stimulus_reaction(observed(provided_result_from_to(E,B,IC))) →
        preparation_for(output(t+c,communicated(send_from_to(B,A,IC))))
present_time(t) ∧
    stimulus_reaction(observed(provided_result_from_to(E,B,IC))) ∧
    not(preparation_for(output(t+c,communicated(send_from_to(B,A,IC))))) →
        stimulus_reaction(observed(provided_result_from_to(E,B,IC)))
    preparation_for(output(t+c,communicated(send_from_to(B,A,IC)))) ∧
    not(output(communicated(send_from_to(B,A,IC)))) →
        preparation_for(output(t+c,communicated(send_from_to(B,A,IC)))) ∧
    present_time(t+c-1) →
        output(communicated(send_from_to(B,A,IC))).

```

For automatic verification of relationships between dynamic properties of different aggregation levels by means of the model checking tool SMV, the obtained finite state transition system is translated into the input format of the SMV model checker. For the description of the translation procedure and the complete SMV specification for the considered example we refer to [11].

One of the possible dynamic properties of the higher aggregation level that can be verified against the generated SMV specification is formulated and formalized in CTL as follows:

GP (Concluding effectiveness): If at some point in time environmental component E generates all the correct relevant information, then later agent C will receive a correct conclusion.

AG (E_output_observed_provide_result_from_to_E_A_info & E_output_observed_provide_result_from_to_E_B_info → **AF** input_C_communicated_send_from_to_A_C_info),

where **A** is a path quantifier defined in CTL, meaning “for all computational paths”, **G** and **F** are temporal quantifiers that correspond to “globally” and “eventually” respectively.

The automatic verification by the SMV model checker confirmed that this property holds with respect to the considered model of the multi-agent system as specified at the lower level.

4 DISCUSSION

The approach to analyzing behavior of a multi-agent system proposed in this paper is based on distinguishing dynamic properties of different aggregation levels and verifying interlevel relations between them using model checking techniques. To enable model checking an automated procedure has been developed, which allows transformation of a behavioral specification of a certain aggregation level first into an executable temporal specification, and subsequently into a description of a finite state transition system. Using this, model checking is performed in the environment SMV. The proposed approach has been evaluated by several case studies, in which verification of agent-based systems from natural domains has been performed.

Notice that an SMV-specification comprises constants, variables and state transition rules with limited expressiveness

(e.g., no quantifiers). Furthermore, for expressing one complex temporal relation a large quantity (including auxiliary) of transition rules is needed. Specification of multi-agent system behavior in the more expressive predicate-logic-based language TTL is much easier. TTL proposes an intuitive way of creating a specification of system dynamics, which still can be automatically translated into a state transition system description, as shown here.

The complexity of the representation of the obtained executable model is linear in size of the non-executable behavioral specification. More specifically, the non-executable specification is related to the SMV specification in the following linear way: (1) for every quantified variable from a non-executable specification a variable and an appropriate rule for its update are introduced; (2) for every nested quantifier an additional variable and an auxiliary executable rule are introduced, which establishes a relation between the quantified variables; (3) for every communicated and observed function from a past and a conditional formulae from dynamic properties, a corresponding memory state creation and a memory state persistence rule are introduced using the variables described in (1) and (2), and variables that correspond to external events; (4) for every non-executable dynamic property auxiliary variables φ_{mem} and φ_{prep} (i.e., the variables that indicate truth values of $\varphi_{mem}(\gamma, t)$ and $\varphi_{prep}(\gamma, t)$ respectively) and corresponding update rules are introduced; (5) for every action and communication specified in $\varphi_{bh}(\gamma, t)$ a variable and an appropriate update rule are introduced; (6) for reactivation of agent preparation states the auxiliary variables and the update rules corresponding to communicated and observed functions from $\varphi_{prep}(\gamma, t)$ are introduced. For verifying an executable model in the SMV OBDD-based symbolic model checking algorithms are used; the study of complexity of such algorithms is given in [9].

REFERENCES

- [1] E.M. Clarke, O. Grumberg, D.A. Peled, *Model Checking*, MIT Press, Cambridge Massachusetts, London England, 1999.
- [2] M. Fisher, Temporal Development Methods for Agent-Based Systems, *Journal of Autonomous Agents and Multi-Agent Systems*, **10**(1), 41-66, (2005).
- [3] J. Hooman, Compositional Verification of a Distributed Real-Time Arbitration Protocol, *Real-Time Systems*, **6**, 173-206, (1994).
- [4] C.M. Jonker and J. Treur, Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness, *International Journal of Cooperative Information Systems*, **11**, 51-92, (2002).
- [5] R. Kowalski and M.A. Sergot, A logic-based calculus of events, *New Generation Computing*, **4**, 67-95, (1986).
- [6] M. Manzano, *Extensions of First Order Logic*, Cambridge University Press, 1996.
- [7] D.A. Marcia, *SADT: Structured Analysis and Design Techniques*, McGraw-Hill, 1988.
- [8] L. Marcio Cysneiros and E. Yu, Requirements Engineering for Large-Scale Multi-agent Systems. In: *Proceedings of the 1st International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS)*, 39-56, (2002).
- [9] K. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.
- [10] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, Cambridge MA: MIT Press, 2001.
- [11] A. Sharpanskykh and J. Treur, *Verifying Interlevel Relations within Multi-Agent Systems: formal theoretical basis*, Technical Report No. TR-1701AI, Artificial Intelligence Department, Vrije Universiteit Amsterdam, (2005). <http://hdl.handle.net/1871/9777>

Flexible Provisioning of Service Workflows

Sebastian Stein and Nicholas R. Jennings and Terry R. Payne¹

Abstract. Service-oriented computing is a promising paradigm for highly distributed and complex computer systems. In such systems, services are offered by provider agents over a computer network and automatically discovered and provisioned by consumer agents that need particular resources or behaviours for their workflows. However, in open systems where there are significant degrees of uncertainty and dynamism, and where the agents are self-interested, the provisioning of these services needs to be performed in a more flexible way than has hitherto been considered. To this end, we devise a number of heuristics that vary provisioning according to the predicted performance of provider agents. We then empirically benchmark our algorithms and show that they lead to a 350% improvement in average utility, while successfully completing 5–6 times as many workflows as current approaches.

1 INTRODUCTION

Computer systems are becoming increasingly complex, distributed and open in nature. In so doing, they pose new challenges to contemporary software engineering techniques, as applications need to deal with many widely distributed and dynamic resources. In this context, service-oriented computing has gained popularity as an appropriate paradigm to build such systems [3]. In this approach, resources are offered as services, which are advertised to consumer applications using computer-readable descriptions and then provisioned dynamically when needed, often as part of complex workflows [2].

A defining feature of these distributed systems is that participants are usually heterogeneous, self-interested agents that act autonomously in accordance with their own private goals [4]. As such, agents choose actions that maximise their own welfare, and so they cannot be assumed to honour every request, to be available whenever they are needed or even to report their capabilities and status truthfully. Additionally, network problems, competition for resources or even software bugs exacerbate the potential for service failures. Thus, when interacting with service providing agents and making real investments that depend on their performance (whether in terms of money, time or other resources), dealing with service failures becomes a vital issue to consider.

Against this background, we believe that flexible service provisioning (i.e., selecting and allocating service providers to specific tasks) is the key to handling and controlling service failures. It allows the consumer to dynamically select service providers based on their performance characteristics and provision replacement providers when services fail. Specifically, provisioning services in the context of a workflow enables a consumer agent to identify particularly failure-prone tasks and invest additional resources in them. To date, however, this provisioning process has received comparatively little

attention. In particular, the fact that a type of service might be offered by multiple providers is seldom explored, and most research is concerned with finding and invoking the first matching service that helps fulfil a given goal [6]. Such a naïve approach means that a single service failure will result in the failure of the whole workflow, which is highly undesirable, especially when success is critical to the interests of the consumer agent.

When it has been considered, this problem is usually addressed reactively by dynamic replanning in case of failure [5], but this method is computationally expensive and can lead to long delays when there are particularly unreliable services. In [1], the authors take a more proactive approach by provisioning services that maximise an expected utility function. However, they again provision only single providers, which makes workflows vulnerable to service failures.

To address these shortcomings, we investigate the process of provisioning service workflows in environments where service success is not generally guaranteed. We show that the current approach of provisioning single providers is insufficient in such environments, and then advance the state of the art by developing several strategies that deal proactively with unreliable providers. This is achieved by provisioning several redundant service providers for a single task in order to reduce the associated failure probability, and by provisioning new providers to those tasks that appear to have failed. We also develop a novel heuristic that provisions workflows in a flexible manner depending on the characteristics of the service providers. Specifically, our heuristic provisions more providers to tasks that are particularly likely to fail, while relying on fewer providers when success is more certain. In order to verify our approach, we report our empirical results that show the value of flexible service provisioning over the naïve approach. In particular, we show that our heuristic successfully completes over 90% of its workflows in most environments where current approaches complete none. Furthermore, the results indicate that our heuristic achieves a 350% improvement in average utility over the naïve strategy.

The remainder of this paper is organised as follows. In the next section we present two simple provisioning strategies, followed by an extended strategy that provisions services in a flexible manner. In Section 3 we describe our experimental testbed and present empirical results for our strategies. Section 4 concludes.

2 PROVISIONING STRATEGIES

In this section, we outline the type of dynamism and uncertainty that service providers display in the complex systems we consider. This is followed by an overview of the kinds of workflows and associated rewards that service consumers typically face. We then present two strategies (*parallel* and *serial*) that provision providers redundantly, but in an inflexible manner, in order to increase their chance of success. Finally, we outline a flexible strategy that provisions providers depending on the agent's assessment of its situation.

¹ School of Electronics and Computer Science, University of Southampton, United Kingdom, {ss04r,nrj,trp}@ecs.soton.ac.uk

2.1 Modelling Service-Oriented Systems

As discussed in Section 1, the inherently uncertain behaviour of autonomous service providers can pose serious threats to the goals of a service consumer. Providers are generally not guaranteed to execute services successfully, and even when they do, the time of completion may be influenced by network traffic, other commitments the provider has made and the hardware a service is executed on. Hence, we model services probabilistically and we assume that some information has already been learnt about the performance characteristics of the service providers for a given task, including their average failure probability and a distribution function for the duration of a successful service execution.

In service-oriented systems, service consumers often face large workflows of inter-dependent tasks. For the purpose of this paper, we represent such workflows as directed, acyclic graphs with the nodes N being tasks and the edges E defining the dependencies between tasks (i.e., an edge $(t_1 \mapsto t_2)$ means that task t_1 has to complete successfully before t_2 can be started). A workflow is only considered complete when all its constituent tasks have been completed.

Furthermore, to evaluate our strategies, we define a utility function $u(t)$ to capture the value of finishing a workflow at time t . We chose to represent this by a maximum utility u_{\max} , awarded for completion, a deadline d , and a penalty charge p , to be deducted from the final utility for every time step a workflow is late. Formally, we write:

$$u(t) = \begin{cases} u_{\max} & \text{if } t \leq d \\ u_{\max} - p(t - d) & \text{if } t > d \text{ and } t < d + u_{\max}/p \\ 0 & \text{if } t \geq d + u_{\max}/p \end{cases} \quad (1)$$

As is common in contemporary frameworks [3], services in our model are invoked *on demand*. Specifically, a consumer requests the execution of a provisioned service when the associated task becomes available. At that time, the consumer incurs a cost (e.g., financial remuneration or communication cost), which is assumed uniform among the providers of a certain task. Thus, the overall profit of a workflow execution is the difference of the utility of finishing the workflow (or 0 if unsuccessful) and its total cost.

Having outlined our environment and basic assumptions, we now continue to develop several provisioning strategies. As discussed, the aim of these is to improve upon the currently predominant strategy of provisioning a single, randomly chosen service provider for each task in a workflow — a strategy that we refer to as the *naïve* strategy.

2.2 Parallel Service Provisioning

Recognising the ease of discovering substitute services in service-oriented systems, our first strategy uses redundant service provisioning to control the effect of unreliable providers. In general, if the success probability of a randomly chosen provider is S_1 , then provisioning n providers for a task results in a success probability S_n :

$$S_n = 1 - (1 - S_1)^n \quad (2)$$

When $S_1 > 0$, then $\lim_{n \rightarrow \infty} S_n = 1$, which implies that it is possible to increase the probability of success to an arbitrarily high amount as long as there are sufficient providers in the system. However, as more providers are provisioned, the total cost incurred rises linearly with n .

This result leads us to our first strategy, *parallel(n)*, which always provisions exactly n randomly chosen members of the set of available providers for a given task. The strategy *parallel(1)*, here, is a special case that is equivalent to the naïve strategy.

2.3 Serial Service Provisioning

An alternative approach to relying on parallel provisioning of providers to increase the probability of success, is to re-provision services when it becomes apparent that a particular provider has failed. In this case, the consumer first provisions a single provider and, after invocation, waits for some time. If the provider has not been successful, the consumer tries a different provider and so on. However, as providers cannot generally be assumed to notify the consumer of failure and because they have non-deterministic duration times, the consumer has to choose an appropriate waiting period. This period should give the provider a reasonable time to finish, but should not waste unnecessary time when the service has already failed².

With this in mind, let $T(w)$ be the probability that a randomly chosen service provider successfully completes within the waiting time w (note that $T(w) \leq S_1$). Then the success probability of serial provisioning with n available providers is:

$$T_n(w) = 1 - (1 - T(w))^n \quad (3)$$

This is usually less than the success probability of provisioning the same number of providers in parallel and the average time taken will also be higher for serial provisioning because of the additional waiting time that is introduced. On the other hand, the average cost drops, because costs are only incurred at the time of invocation.

This leads us to our second strategy, *serial(w)*, which always provisions exactly one randomly chosen member of the set of available providers. After a waiting period of w time units, if no success has been registered yet and if there are still more providers, the agent re-provisions a new, randomly chosen provider. A special case of this strategy, *serial(∞)*, is equivalent to the naïve strategy.

2.4 Flexible Service Provisioning

The strategies discussed so far provision services in an inflexible manner, as they always select the same number of providers for each task. This approach is insufficient in most scenarios, however, because some services may benefit from being provisioned redundantly, while others have a high degree of certainty and so need not be provisioned in this manner. Furthermore, it is not clear how to choose n and w in the above strategies so as to balance the associated cost and maximise the expected utility of a workflow, and it is desirable to automate this decision.

To address these shortcomings, in this section we develop a *flexible* strategy that combines the above two strategies, *parallel(n)* and *serial(w)*, by automatically finding values for the number of parallel invocations (n_i) and waiting time (w_i) for every task t_i in a given workflow. To focus on the basic problem, we assume that this allocation is made once and then used non-adaptively for the entire execution of the workflow³. We also assume that the consumer is risk-neutral — that is, we want our algorithm to choose values n_i and w_i in order to maximise the consumer's long-term expected profit.

Now, many optimisation approaches exist in the literature, but due to the complex nature of this particular problem, which includes the

² When re-provisioning services, we assume that all previously provisioned services are subsequently ignored (even if they had succeeded at a later time). This assumption draws a clear distinction between serial and parallel provisioning techniques, but does not fundamentally alter our results.

³ This simplifies the problem and helps us verify that our algorithm makes good initial predictions even without receiving ongoing feedback during a workflow. However, in future work we will use such information to update the agent's service provisions dynamically.

summation of random variables (the service durations), integer variables and a non-linear objective function, we have decided to use a local search technique that approximates the expected profit using a heuristic function, and searches for a good allocation rather than an optimal one.

Specifically, our local search algorithm begins with a random allocation and then applies steepest ascent hill climbing [7] to gradually improve the allocation until a maximum is found. The heuristic function we use is based on the utility function u and we use it to calculate the estimated profit \tilde{u} :

$$\tilde{u} = p \cdot u(\tilde{t}) - \bar{c} \quad (4)$$

where p is the probability of finishing all tasks at some point in time, \tilde{t} is the estimated completion time for the workflow, and \bar{c} is the expected cost.

In the following, we explain how p , \tilde{t} and \bar{c} are calculated, starting with appropriate calculations for each individual task in Section 2.4.1 and then extending this to the whole workflow in Section 2.4.2.

2.4.1 Local Task Calculations

We are interested in three performance measures for each task t_i in the workflow — the overall success probability of the task (s), its expected cost (\bar{c}) and its expected duration (\bar{d}). These are calculated using the information that the consumer has about each task and the relevant providers:

- p is the number of providers provisioned in parallel for the task.
- w is the waiting time before a new set of providers is provisioned.
- a is the number of available providers.
- f is the average failure probability of a provider.
- c is the cost of a provider.
- D is the cumulative density function for a single service duration.

First, we calculate the probability of success s as in equation 3, but noting that $T = (1 - f) \cdot D(w)$ and using the total number of available providers:

$$s = 1 - (1 - (1 - f) \cdot D(w))^a \quad (5)$$

In showing how to calculate the total cost, we assume that $a \bmod p = 0$ (i.e., that we can invoke up to $m = a/p$ sets of p providers with no remaining providers at the end⁴). Each invoked set of providers then has a probability of success $\hat{s} = 1 - (1 - (1 - f) \cdot D(w))^p$ and an associated cost cp . The consumer is guaranteed to pay cp at least once, and may pay again if the previous invocation was unsuccessful with probability $\hat{f} = (1 - \hat{s})$. Using this, and assuming that $\hat{f} < 1$, we can write the expected cost \bar{c} as:

$$\bar{c} = cp \sum_{k=0}^{m-1} \hat{f}^k = cp \frac{1 - \hat{f}^m}{1 - \hat{f}} \quad (6)$$

Using the same assumptions and treating the simpler case with $a \bmod p = 0$, we now investigate how to calculate the expected duration of a task, \bar{d} . We define this to be the mean time until the first provider carries out the task successfully, conditional on an overall success (i.e., at least one provider is successful).

First, we define μ to be the mean duration of a single successful invocation of p providers (also conditional on overall success). Then we follow a similar technique for calculating \bar{d} as we did for \bar{c} . We consider all possible outcomes for the invocation by calculating the expected duration when the task is completed after exactly k failed

⁴ Due to space restrictions in this paper, we omit the general case.

invocations as well as the associated probability. Then we multiply these durations with their probabilities and sum them to get the overall expected duration.

More formally, if a task succeeds after exactly k unsuccessful invocations, the expected duration, \bar{d}_k , is the expected duration for a single invocation added to k waiting time durations w for the previously failed invocations:

$$\bar{d}_k = \mu + kw \quad (7)$$

The probability of the task completing after the k th attempt is:

$$s_k = \hat{f}^k (1 - \hat{f}) \quad (8)$$

With this, we calculate the overall expected duration, conditional on at least one provider being successful (assuming $\hat{f} < 1$):

$$\begin{aligned} \bar{d} &= \frac{1}{s} \cdot \sum_{k=0}^{m-1} \bar{d}_k s_k \\ &= \frac{1}{s} \cdot \sum_{k=0}^{m-1} ((\mu + kw) \cdot \hat{f}^k (1 - \hat{f})) \\ &= \frac{1}{s} \cdot \left(\mu (1 - \hat{f}^m) + w \frac{\hat{f} - m\hat{f}^m + (m-1)\hat{f}^{m+1}}{1 - \hat{f}} \right) \end{aligned} \quad (9)$$

We have now shown how to calculate various performance characteristics of a single task. In the following section, we explain how this is extended to calculate the overall heuristic function for an allocation over the whole workflow.

2.4.2 Global Workflow Calculations

Let s_i , \bar{c}_i and \bar{d}_i be the success probability, the expected cost and the expected duration of task t_i . With this information for each task, we are now interested in calculating the overall probability of success s , the estimated completion time of the workflow \tilde{t} and the total expected cost \bar{c} .

The overall probability of success is simply the product of all s_i :

$$s = \prod_{\{i | t_i \in N\}} s_i \quad (10)$$

The expected total cost is the sum of all task costs, each multiplied by the respective success probabilities of their predecessors in the workflow (where r_i is the probability that task t_i is ever reached):

$$\bar{c} = \sum_{\{i | t_i \in N\}} r_i c_i \quad (11)$$

$$r_i = \begin{cases} 1 & \text{if } \forall t_j \cdot ((t_j \mapsto t_i) \notin E) \\ \frac{1}{\prod_{\{j | (t_j \mapsto t_i) \in E\}} s_j} & \text{otherwise} \end{cases} \quad (12)$$

Finally, we approximate the overall time \tilde{t} using the length of the critical path in the workflow. This is the length of the longest path from any node with no predecessors to any node with no successors, using the expected durations \bar{d}_i as weights attached to the nodes. Such an approach will normally underestimate the true expected duration, because it focusses only on one path, ignoring the possibility that other services outside this path may take longer. However, it provides a fast approximation and has been used widely in project management and operations research [8]. Formally, we let $P = \{t_i \mid t_i \text{ is on the critical path}\}$. This gives us:

$$\tilde{t} = \sum_{\{i | t_i \in P\}} \bar{d}_i \quad (13)$$

Using these values and the heuristic function given in equation 4, it is now possible to use steepest ascent hill-climbing to find a good allocation of providers. In practice, we found it useful to perform this in two iterations — once using a modified utility function \tilde{u} , which is a linear version of u , giving a higher reward than usual for finishing early and a larger loss for finishing late, and then again using the normal utility function u . This approach allows the algorithm to escape from a common local maximum where the agent decides to concede, allocate minimal resources to the tasks and hence incur a low net loss, but also a very low probability of success. Furthermore, we could generally increase performance by adding a constant amount of extra time to \tilde{t} to account for the error in the prediction. In all our experiments we set this to 20% of the workflow deadline, as this produces good results in a variety of experimental settings.

3 EMPIRICAL EVALUATION

As stated in Section 1, the aim of our work is to deal effectively with unreliable service providers when provisioning workflows. To this end, in this section we empirically compare our proposed strategies to the currently predominant naïve approach. In particular, we investigate the average utility gained by all strategies, as well as the average proportion of successfully completed workflows. In the remainder of this section, we describe our experimental testbed and methodology, followed by the results.

3.1 Experimental Testbed

In order to analyse our strategies empirically, we developed a simulation of a simple service-oriented system. In this simulation, we generate large numbers of random workflows and measure the performance of our strategies by recording the percentage of workflows that failed (where t time steps had elapsed, so that $u(t) \leq 0$) and succeeded (where all tasks were completed within t time steps, so that $u(t) > 0$). We also measure the average profit of a single service consuming agent.

Our simulation is discrete, notifying the consumer of any successful service execution once every integer time step, at which point new services are also invoked according to the consumer's provisioning strategy. While the consumer is notified in case of service failure, no information is given when a provider fails.

To simplify the analysis and because our approach does not deal directly with differentiating between individual providers at this time, we examined environments with homogeneous service providers (i.e., all providers share the same success probability and duration distributions).

For the data presented in this section, we used workflows consisting of 10 tasks in a strict series (i.e., without parallel tasks, because this allowed us to verify some results analytically), we assumed that there were 1,000 providers for every task with each provider having a cost of 10 and a gamma distribution with shape $k = 2$ and scale $\theta = 10$ as the probability distribution of the service duration. We set a deadline of 400 time units for the workflow, an associated maximum utility of 1,000 and a penalty of 10 per time unit. We also performed similar experiments in a variety of environments, including heterogeneous and parallel tasks, and observed the same broad trends that are presented in the following section.

To prove the statistical significance of our results, we averaged data over 1,000 test runs and performed an analysis of variance

(ANOVA) where appropriate to test whether the strategies we tested produced significantly different results. When this was the case, we carried out pairwise comparisons using the least significant difference (LSD) test. Thus all results reported in this paper are statistically significant ($p = 0.001$).

3.2 Experimental Results

In our first experiment, we compared the performance of strategy $\text{parallel}(n)$ ⁵ with the naïve approach in environments where service providers have a varying probability of failure (see Figure 1). From this, it is clear that there is a considerable difference in performance between the different strategies — the average profit gained by the naïve strategy falls dramatically as soon as failures are introduced into the system. In this case, the average utility of provisioning single providers falls to below 0 when the failure probability of providers is only 0.3. A statistical analysis reveals that the naïve strategy dominates the other two when there is no uncertainty in the system. However, as soon as the failure probability is raised to 0.1, $\text{parallel}(2)$ begins to dominate the other strategies. Between 0.3 and 0.6 $\text{parallel}(6)$ then becomes the dominant strategy as increased service redundancy leads to a higher probability of success. Above this, the parallel strategies do not yield better results than the naïve strategy as they also begin to fail in most cases.

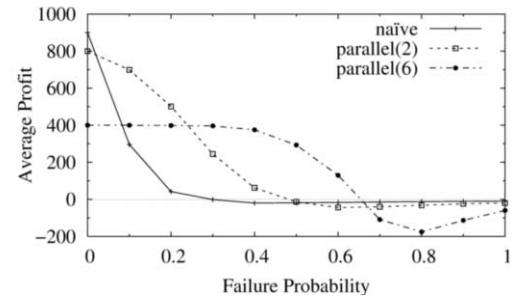


Figure 1. Effect of provisioning different numbers of providers in parallel

Summarising these trends, it is obvious that redundant provisioning yields a considerable improvement over the naïve approach in a range of environments. For example, when the failure probability is 0.2, provisioning two providers results in an almost 1,100% improvement in average profit over the naïve strategy. However, no redundant strategy dominates the other and both eventually make losses when the probability of failure increases to such an extent that the chosen redundancy levels do not suffice to ensure success.

We carried out a similar experiment to verify the advantage of serial provisioning over the naïve strategy (see Figure 2). Here, again, there is a marked improvement over the naïve strategy for failure probabilities up to and including 0.5. This improvement is due to the fact that serial provisioning responds to failures as they occur, while only paying for additional services when necessary. However, as the failure probability rises, this strategy begins to miss its deadlines and hence incurs increasingly large losses.

Finally, to show how the *flexible* strategy compares against the naïve provisioning approach and inflexible redundancy, Figure 3 shows our experimental data (again, using the same experimental variables). The top graph shows the percentage of workflows that

⁵ Here, we arbitrarily chose $n = 2$ and $n = 6$ as representative of the general trends displayed by the strategy as more providers are provisioned.

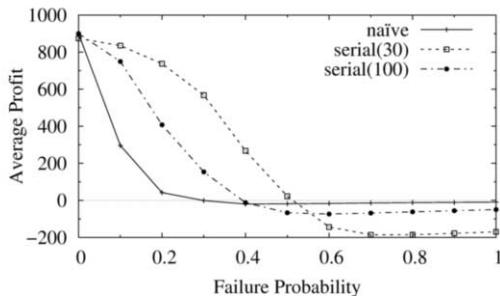


Figure 2. Effect of different amounts of waiting times for re-provisioning

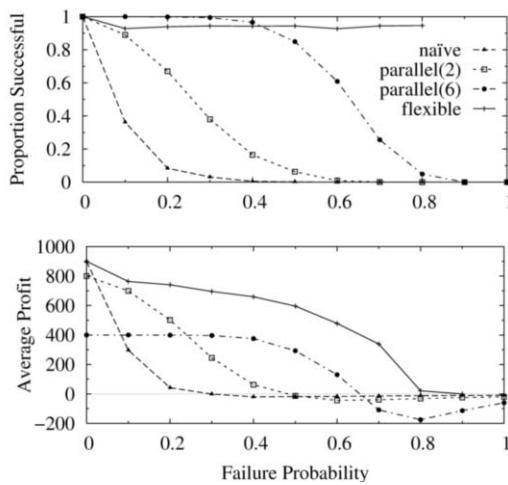


Figure 3. Performance of flexible strategy

succeeded out of all the ones generated. Here, we see that our heuristic approach initially performs slightly worse than *parallel(6)* (but always significantly better than the naïve approach). This is due to our technique of using the critical path of the workflow as an estimate for the total time taken. This technique is usually too optimistic and might result in under-provisioned tasks. However, the graph clearly shows that the flexible technique still achieves a success-rate of over 90% and, more importantly, maintains this up to a failure probability of 0.8, by which all other approaches have large failure rates. When 0.9 is reached, the strategy begins to ignore all workflows, because it cannot find a feasible allocation to offer a positive return. On the lower graph, we show the average utility that is gained by the same strategies. Here, it is clear that the flexible approach performs better than any of the other strategies. This is due to its utility prediction mechanism and the fact that it can make choices separately for the various tasks in the workflow. This flexibility allows the strategy to provision more providers for latter parts of the workflow, where success becomes more critical as a higher investment has already been made. The flexible approach also combines the benefits of the other strategies, allowing the agent to choose between parallel (e.g., when there is little time) and serial provisioning (e.g., when the agent can afford the extra waiting time) or a mixture of the two. Although performance degrades as providers become more failure-prone, flexible provisioning retains a relatively high average utility when all other strategies start to make a loss. Furthermore, the strategy avoids making a loss due to its prediction mechanism, which ignores a workflow when it seems infeasible.

To conclude, Table 1 summarises the performance of some rep-

Table 1. Summary of results with 95% confidence intervals

Strategy	Mean profit	Profit vs naïve	Success rate
naïve	103.09 ± 5.30	1	0.13 ± 0.01
parallel(6)	175.87 ± 5.79	1.71 ± 0.20	0.61 ± 0.01
serial(30)	221.91 ± 7.92	2.15 ± 0.26	0.44 ± 0.01
flexible	471.99 ± 8.71	4.58 ± 0.49	0.77 ± 0.01

resentative strategies, averaged over all environments that we tested (using the same data as in Figures 1–3). These results highlight the benefits of our strategies, and show that our flexible strategy by far outperforms the naïve approach. In particular, we achieve an improvement of approximately 350% in mean profit and successfully complete 76–78% of all workflows.

4 CONCLUSIONS

In this paper, we addressed the problem of dealing with unreliable service providers in complex systems. To this end, we developed a novel algorithm for provisioning workflows in a flexible manner and showed that this algorithm achieves a high success probability in uncertain environments and that it outperforms the current, naïve strategy of provisioning single providers. This work is particularly relevant for distributed applications where several services are composed by a consumer agent, and where these services are offered by autonomous agents whose success cannot be guaranteed. Important application domains for our approach include scientific data processing workflows and distributed business applications, where services are sourced from across departments or organisations.

In future work, we plan to extend our approach to cover more heterogeneous environments, where service providers differ significantly in terms of cost and reliability within the same service type and where such qualities are affected dynamically by system-wide competition and resource availability. Furthermore, we plan to examine in more detail the computational cost of provisioning and how this can be balanced in a flexible manner with the need to make quick decisions in dynamic environments.

ACKNOWLEDGEMENTS

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) and a BAE Systems studentship.

REFERENCES

- [1] J. Collins, C. Bilot, M. Gini, and B. Mobasher, ‘Decision Processes in Agent-Based Automated Contracting’, *IEEE Internet Computing*, **5**(2), 61–72, (2001).
- [2] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda, ‘Mapping Abstract Complex Workflows onto Grid Environments’, *Journal of Grid Computing*, **1**(1), 25 – 39, (2003).
- [3] M. N. Huhns and M. P. Singh, ‘Service-Oriented Computing: Key Concepts and Principles’, *IEEE Internet Computing*, **9**(1), 75–81, (2005).
- [4] N. R. Jennings, ‘On Agent-Based Software Engineering’, *Artificial Intelligence*, **117**(2), 277–296, (2000).
- [5] M. Klusch, A. Gerber, and M. Schmidt, ‘Semantic Web Service Composition Planning with OWLS-XPlan’, in *Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web*, (2005).
- [6] S. A. McIlraith and T. C. Son, ‘Adapting Golog for Composition of Semantic Web Services’, in *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, pp. 482–493, Toulouse, France, (2002).
- [7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2nd edn., 2003.
- [8] W. L. Winston, *Operations Research: Applications and Algorithms*, Wadsworth Publishing Company, 3rd edn., 1997.

Heuristic Bidding Strategies for Multiple Heterogeneous Auctions

David C. K. Yuen,¹ Andrew Byde,² Nicholas R. Jennings¹

Abstract. This paper investigates utility maximising bidding heuristics for agents that participate in multiple heterogeneous auctions, in which the auction format and the starting and closing times can be different. Our strategy allows an agent to procure one or more items and to participate in any number of auctions. For this case, forming an optimal bidding strategy by global utility maximisation is computationally intractable, and so we develop two-stage heuristics that first provide reasonable bidding thresholds with simple strategies before deciding which auctions to participate in. The proposed approach leads to an average gain of at least 24% in agent utility over commonly used benchmarks.

1 Introduction

The growing number of online auction sites liberates individual shoppers from searching only within their surrounding community. Moreover, if the item is popular, it is not uncommon to find hundreds of auctions offering the same good³. However, it can be difficult for a human to keep track of more than a few auctions. But, *global bidders* that can accept information and participate in a large number of auctions are likely to receive extra benefit when compared with bidders that only operate locally in one auction [8]. When all of these factors are taken together, it is apparent that there is a growing need to develop autonomous agent bidding strategies that can operate over multiple auctions.

To this end, we seek to devise a best response bidding strategy for global bidders participating in multiple auctions. Each such bidder may have a demand for one or more units of identical private-value goods. Our primary interest is in multiple heterogeneous auctions, because it is the most prevalent case in practice, in which the auction format and the starting and closing times can be different. We assume each seller auctions one item and these items are perfect substitutes. Due to its complex nature, most analytical studies of heterogeneous multiple auctions are based on the simplified case of homogeneous auction types that either operate sequentially [9] or completely in parallel [8]. For our more general setting, heuristic approaches are the norm [1, 2, 3, 4, 5] and are the approach we adopt here.

Now, to develop a bidding strategy that maximises the expected utility, it is theoretically possible to model the multiple auctions as a Markov Decision Process (MDP) and calculate the expected utility by backward induction [2, 3]. The utility evaluation can be simplified by adopting pessimistic estimation [4], in which an agent sets its bids using no information on the ordering of the future auctions. However, issues associated with computational intractability limit these approaches to only a low number of auctions. In contrast, Anthony *et al.* [1] devised a heuristic bidding strategy that combines the effects of the desire to obtain a bargain, the deadline and the desperation for obtaining an item into a single formula. However, their bidding tactic ignores the fact that some of the remaining auctions can be more favourable to the global agent (e.g. have fewer local bidders). Relatedly, the bidding strategy developed by Dumas *et al.* [5] consists of a probabilistic bid learning and planning stage. However, their algorithm considers simultaneous auctions as incompatible and would bid in only one of them.

Against this background, this paper develops a two-stage heuristic approach to approximate the best response bidding strategy for a global bidder. In the first stage, a threshold heuristic is employed to compute a maximum bid or threshold for each auction. Then in the second stage, the agent decides whether it should participate in each of the available auctions using an auction selection heuristic that exploits the bidding thresholds calculated from the first stage. A number of threshold and auction selection heuristics are developed under this flexible framework. In developing these heuristics, this work advances the state of art in a number of ways. First, we devise several two-stage heuristics that enable the agent to improve its utility further by bidding in more auctions than its demand. This tactic requires the agent to manage the excess procurement risk carefully and is an area ignored by previous bidding heuristics other than those using MDPs or exhaustive search approaches. Second, we provide a pseudo-polynomial time auction selection heuristic that closely approximates the near optimal solutions of an exponential time exhaustive search algorithm from the literature [4]. Third, we extend the use of equal bidding thresholds to maximise the agent utility from complementary goods [8] to our perfect substitute setting and verify that the agent's utility received is very close to the global maximum (at least for simultaneous second-price auctions).

The paper is structured as follows. Section 2 details the heuristics applied to set the threshold and select the auctions to participate in. Section 3 empirically evaluates the strate-

¹ Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK. {dy,nrj}@ecs.soton.ac.uk

² HP Labs, Filton Road, Stoke Gifford, Bristol, BS34 8QZ, UK. andrew.byde@hp.com

³ To illustrate the scale of this problem, within eBay alone, more than a thousand auctions were selling Apple's 4GB iPod mini at the time of writing.

gies and compares them with other heuristics proposed in the literature. Section 4 concludes.

2 The Multiple Auction Bidding Strategy

Before detailing the bidding heuristics, we formalise the multiple auction setting being studied. Here an auction a can be characterised by a tuple $(t^s(a), t^c(a), N(a), \theta(a))$ consisting of the starting time, closing time, number of local bidders and auction format for auction a . Notice that since $N(a)$ is the number of *local* bidders in an auction; the total number (including the agent itself) is $N(a) + 1$. The auction format can be one of the standard four single-sided types, English, Dutch, first-price (FPSB) or second-price sealed bid (SPSB), denoted as {ENG, DUT, FPSB, SPSB}. We introduce the degree of overlap $\lambda \in [0, 1]$:

$$\begin{aligned} \text{time with at least } 1 \text{ running auction: } T_{occ} &= |\{t \in \mathbb{N} | \exists a : t^s(a) \leq t < t^c(a)\}| \\ \text{degree of overlap: } \lambda &= \frac{\sum_a (t^c(a) - t_s(a)) - T_{occ}}{T_{occ} (M - 1)} \end{aligned}$$

to characterise a set of M auctions by the starting and closing times. Purely sequential and purely simultaneous auctions are the two extreme cases with $\lambda = 0$ and 1 respectively.

The agent's valuation V for obtaining multiple items is linear with random coefficient v per item up to a maximum of k items: the agent derives value $V(x) = v \min(k, x)$ from x items, and the utility of the agent for obtaining x items at cost Π is quasi-linear: $U(x, \Pi) = V(x) - \Pi$.

We assume that all bidders have identically distributed independent item-valuations v with distribution $F(x) = \text{Prob}(v \leq x)$, $f(x) = F'(x)$. We will use the uniform distribution in $[0, 1]$ as an example throughout the text, but the heuristics themselves are equally applicable to other distributions unless specified otherwise. For a set of M auctions, our goal is to devise a bidding strategy $b = (b_1, b_2, \dots, b_M)$ where b_a gives the threshold (i.e. the maximum bid that should be placed) for auction a . In sealed bid auctions, the threshold is the bid placed, in Dutch auctions, the agent bids when the current price falls the threshold, and in English auctions, the agent bids until the current price reaches the threshold. The threshold is set to zero if the auction selection heuristic indicates that the agent should not participate in that auction.

At any point in time, the bidding strategy considers the subset of auctions A^{av} that is *available*. An auction is *unavailable* if it is either closed or is of type ENG and the current price is already higher than the agent's threshold. We use $M^{av} = |A^{av}|$ for the number of available auctions. The agent is expected to keep track of the number of acquired items. At time T , the agent has an extra capacity of k_T items = $k - x_T$ where x_T is the number of items already acquired.

2.1 The Threshold Heuristics

The threshold set required by the auction selection stage is generated using one of the heuristics developed in this section. In all cases, the setting of the threshold is affected by the differences between individual auctions (e.g. the number of bidders and the auction format) and by the availability of the item in future auctions. Since we want to identify the most influential factors to the received utility, each of our heuristics examines only a subset of the above factors. Specifically, we

develop two heuristics. The *single auction dominant* heuristic assigns different thresholds according to the auction format and the number of bidders in each auction. The *equal threshold* heuristic assigns the same threshold to every auction and adjusts this according to the availability of the item in future auctions. Some heuristics in the literature have also examined the bidding history [5] and user preferences [1] when adjusting the bidding level. But, with our modular two-stage heuristic design, these external factors can easily be included later by extending or replacing the existing heuristics.

2.1.1 The Single Auction Dominant Heuristic

This heuristic sets a different threshold for each of the auctions according to the auction format and the number of bidders. Specifically, the threshold is assigned as if it were the only auction available and the agent had valuation v for a single item. We choose the single auction dominant bidding heuristic not only because its analytical solution is well known [7], but also because of its ability to adjust the bid for different auction types and numbers of bidders with little computational overhead. The threshold is the same as the true value for second price mechanisms (ENG, SPSB) and for the first price mechanisms (DUT, FPSB), it becomes [7]:

$$b_a = v - F^{-N(a)}(v) \int_0^v F^{N(a)}(x) dx$$

which is $v N(a)/(N(a)+1)$ if the private values of the bidders follow a uniform distribution.

2.1.2 The Equal Threshold Heuristic

This heuristic sets a single threshold, b_{eq} , for all auctions. This choice is justified because the global bidder's utility in multiple auctions selling complementary goods can be maximised with equal-bid pairs [8] and our brute-force best bidding strategy search upon simultaneous auctions with unit agent demand ($k = 1$) seems to suggest that placing equal bids is also suitable for our perfect substitute setting⁴.

Our goal is to find the equal bid threshold that approximately maximises expected utility:

$$\begin{aligned} E(U|b) &= E(V|b) - E(\Pi|b) \\ &= vE(\min(k, X)|b) - \sum_{a \in A^{av}} E(\Pi(a)|b) \end{aligned} \quad (1)$$

To do this, we adopt a procedure that estimates the likely number of units won (X) and the likely payments (Π) separately. Now, the expected payment $E(\Pi|b)$ is the sum of payments for each auction and the expected payment $\Pi(a)$ in a particular auction a from using threshold $b(a)$ is:

$$E(\Pi(a)|b(a)) = \int_0^{b(a)} y F^{N(a)-1}(y) f(y) dy \quad (2)$$

For valuations v that are uniformly distributed in $[0, 1]$ this gives an expected payment of $N(a) b^{N(a)+1}/(N(a) + 1)$. However, the valuation term is non-linear with respect to the number of units won, and so the situation is more complicated.

⁴ It should be noted that the optimality of equal bids is not universal even for simultaneous SPSB auctions with identical bidder distributions. For example, in the extreme case in which the valuation approaches the upper range, it is best to bid one's true value for k requested items and zero for the rest.

Thus to estimate the number of units the agent might win by using a particular bid threshold, we replace the available current and future auctions with simultaneous SPSB auctions. We believe this is a reasonable assumption for two reasons. First, the expected revenue of second-price auctions is only slightly lower than those of first-price ones. Second, unlike currently running English (or Dutch) auctions in which we know the closing price must be larger (or smaller for Dutch) than the current price, we do not have such information regarding future auctions. So, in terms of information availability, it resembles the situations in simultaneous sealed bid auctions.

Now, to measure the number of bidders \bar{N} in the set of available auctions A^{av} we use the harmonic mean $\frac{1}{\bar{N}} = \frac{1}{M^{av}} \sum_{a \in A^{av}} \frac{1}{N(a)}$ because the agent is less likely to win in auctions with many bidders and so should give less weight to their consideration.

In so doing, we assume that the events of winning each auction are independent and identically distributed, so that the number of items the agent will win has a binomial distribution. The probability of an agent with threshold b winning in any second-price auction with n other local bidders (who will bid their valuation) is $p(b) = F^n(b)$. The probability of winning x items out of M^{av} auctions is:

$$\text{Prob}(X = x|b) = \binom{M^{av}}{x} p(b)^x (1 - p(b))^{M^{av}-x}$$

Combining this with:

$$E(\min(k, X)|b) = \sum_{x \leq k} x \text{Prob}(X = x|b) + k \text{Prob}(X > k|b)$$

allows us to estimate the value term in (1).

Since the agent is not compensated for any extra units acquired, its utility drops rapidly if the purchase quota is exceeded accidentally. However, to identify the equal threshold b_{eq} that approximates the maximum utility, a simple one dimensional *golden section search* [10] is sufficient because the utility curve has only a single maximum.

2.2 The Auction Selection Heuristics

The auction selection heuristics determine which available auctions the agent should participate in, on the basis of the collection of bidding thresholds calculated using a threshold heuristic. With our flexible architecture, it is possible to introduce heuristics that satisfy different user preferences [1] at a later stage. But for now, we intend to develop heuristics that approximate well to the global utility maximum, which is the goal for most users. Specifically, we describe two such heuristics: an *exhaustive search* heuristic inspired by [4], and a much more computationally tractable *knapsack utility approximation* heuristic.

2.2.1 Exhaustive Search Selection

Byde et al. [4] describe an exhaustive search procedure for bid decision making. Their algorithm subsumes the selection of auctions into the choice of thresholds: ideally all tuples of thresholds would be tested, which includes the “trivial” zero thresholds that are equivalent to not participating in an auction at all. In our context, where thresholds are determined by a separate process, exhaustive search consists of testing

almost all subsets of available auctions. Nevertheless, some heuristic properties of different auction formats are exploited to prune the search space. For example, in English auctions, we assume that an agent prefers to bid in auction a_1 if its current price is lower than that of a_2 while the expected chance of winning is higher. In sealed bid auctions, the agent refrains from submitting a bid until the deadline is imminent so that it can learn from more auctions before making a decision.

The expected value $E(V|S, b)$ of bidding only in a set of auctions S is calculated using the sum over all possible sets $W \subseteq S$ of auctions that might be won:

$$E(V(X)|S, b) = \sum_{W \subseteq S} \left(V(|W|) \prod_{a \in W} p_a(b(a)) \prod_{a \notin W} (1 - p_a(b(a))) \right) \quad (3)$$

where $p_a(b(a))$ is the probability of winning auction a given threshold $b(a)$. Expected payments are given by (2) as above.

Clearly this heuristic is expensive to calculate both because of the exhaustive search over the set of auctions S in which to bid, and because of the exponential complexity of (3).

2.2.2 Knapsack Utility Approximation

The development of this heuristic is inspired by our observation that the utility is non-linear with respect to the number of wins. Specifically, each additional acquired item provides extra utility until the maximum demand k is reached. Beyond that point, each additional item incurs a large marginal loss because the agent does not receive any extra compensation for the increased payment. Now, since the number of wins is a random variable and the utility plummets if the agent acquires more than k items, it is crucial to carefully control the risk of exceeding the demand limit. Given this, the rationale for the heuristic proposed is to identify a subset of available auctions that minimises the expected payment (i.e. locally maximises the utility), while maintaining the expected number of wins within a certain safe level. The knapsack algorithm was chosen because its solution can be approximated in pseudo-polynomial time by dynamic programming [6].

In more detail, the heuristic works as follows. First, the agent (re)-evaluates its capacity for extra items. For planning purposes, the extra capacity $\hat{k}_T = k_T - H$ should be reduced by the number of holding auctions H . A holding auction is any sealed bid auction that the agent has submitted a bid to and has yet to receive the results or any English auction in which it is currently leading. The intuition here is that the agent has a chance to win once it holds a bid in an auction. Therefore, it should temporarily reduce the demand limit to prevent accidentally exceeding its purchase quota. Second, by making a similar argument to that of Section 2.1.2 when we simplified the auction setting, we model the multiple auctions as if they are all SPSB and run simultaneously. Given this, the threshold set generated by the threshold heuristics is converted into a M^{av} -tuple $b = (b(a_1), b(a_2), \dots, b(a_{M^{av}}))$ with its first element corresponding to the threshold of the auction with the least number of bidders, and so on. If the agent selects n auctions to participate in, it will submit n non-zero bids out of a choice of M^{av} assuming the agent has a preference for auctions with fewer bidders. The expected utility is calculated for $k \leq n \leq M^{av}$. The number of auctions to participate in,

n_{opt} , that gives the highest expected utility is then identified:

$$b_n = (\underbrace{b(a_1), b(a_2), \dots, b(a_n)}_n, \underbrace{0, \dots, 0}_{M^{av} - n})$$

$$n_{opt} = \arg \max_{n \geq k_T} E(U|b_n) \quad (4)$$

Now, the agent could just decide to participate in the n_{opt} auctions with the least number of bidders. However, this is not likely to be a good enough strategy because the simplified model ignores the differences between auction formats and discards a lot of useful information obtained after the auction starts (such as the current price of the English and Dutch auctions). On the other hand, with the exception of the case when M is small, it is impractical to calculate the exact utility as it requires the probability calculation to be repeated for each of the 2^M possible winning combinations. In contrast, if we assume the outcomes of the auctions are independent, the expected number of wins can easily be calculated as the sum of the winning probabilities of the individual auctions. By maintaining the expected number of wins at a sufficiently low level, we can mitigate the risk of accidentally purchasing too many items even if we place more than k_T bids.

We now turn to the issue of finding the optimal expected number of wins. Assuming the agent bids in n_{opt} auctions with threshold $b_{opt} = (b(a_1), b(a_2), \dots, b(a_{n_{opt}}), 0, \dots, 0)$, the expected number of wins that gives highest utility⁵ can be calculated as:

$$\hat{x}_{opt} = E(X|b_{opt}) = \sum_{i=1}^{n_{opt}} p_a(b(a_i)) + x_T$$

Finally, we can apply a knapsack algorithm to find the combination of auctions that minimises the payment, while keeping the predicted number of winning items less than or equal to the optimal number of winning items \hat{x}_{opt} . Generally speaking, the goal of the knapsack algorithm is to maximise the combined value of items packed into a bag, providing the size constraint has not been violated. Here the optimal winning number \hat{x}_{opt} is taken as the weight constraint and each auction in the subset A^{av} is a candidate item available for selection. For auction a , the weight is the chance of winning $p_a(b_a)$ if bidding with the threshold $b(a)$ found by the heuristics in Section 2.1. The value for participating in auction a is defined as minus the expected payment, so that maximising value minimises payment. The knapsack algorithm sets the bidding thresholds to zero to indicate the auctions that the agent should not participate in. The best selection of auctions to participate in is re-evaluated at each time step.

The expected payment is given by (2), but, for current English (or Dutch) auctions, the expected payment is set to the current price x_t plus (or minus for Dutch auction) the minimum bid increment. It is because the current price update at each round presents a prime opportunity for the agent to hunt for bargains. For example, if there is a more favourable auction later in which the agent is planning to bid up to $b(L)$ with an expected payment of $\pi(L)$, there is no harm in bidding lower than $b(L)$ for the current English or Dutch auction. It may get an even better deal by chance.

⁵ It is important to note that the expected number of wins $E(X)$ is different from the upper demand limit k . For example, the expected number of wins can be much smaller than the upper demand limit if the buyer has a low valuation.

3 Empirical Evaluation

This section evaluates the heuristics we have devised. We have introduced two threshold heuristics – single auction dominant (DOM) and equal threshold (EQT), and two auction selection heuristics – exhaustive search selection (ES) and knapsack utility approximation (KS). In particular, we are interested in evaluating how the combined heuristics (DOM-ES, DOM-KS, EQT-ES, EQT-KS) perform. As benchmark, we use a random strategy (RND) that chooses k auctions at the start then bids locally in them and a greedy strategy (GRD) that participates in the k_T auctions with the least number of bidders at time T . Both RND and GRD place bids following the dominant strategy for a single auction. In our experiment, we consider two scenarios: (1) simultaneous SPSB auctions, (2) unrestricted settings, in which the number of bidders, the auction types, and the starting and closing times are all randomised. The first scenario is considered because simultaneous SPSB auctions require only one round of decision. Thus this simplified setting allows us to search for the optimal bidding strategy using a global maximisation technique⁶ and compare that with our heuristics. The second scenario is actually our focus because it is the one that most closely represents reality.

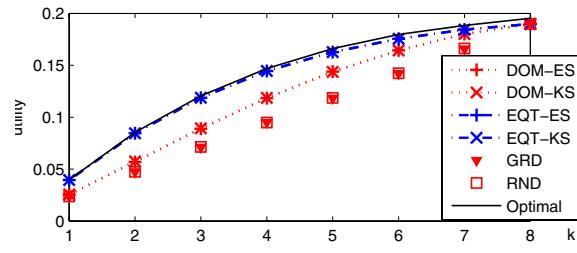


Figure 1. Agent utility for simultaneous SPSB auctions ($M = 8, N = 5$).

The first environment evaluates the agent utility for a set of 8 simultaneous SPSB auctions. Specifically, Figure 1 shows the average utility as agent demand k varies from 1 to 8. As can be seen, the combined heuristics using EQT (EQT-ES and EQT-KS) perform better than those using DOM and are within 3% of the optimal. Now, the DOM heuristic adjusts the bidding level in accordance to factors affecting each single auction (i.e. number of bidders and auction format), whereas EQT considers the future item availability from all the remaining auctions. Thus the latter effect dominates, especially for larger M , and so explains the observed improvement when using EQT. Although DOM is less successful, by using our ES or KS decision heuristics, 15-25% of the utility loss can still be recovered when compared with either GRD or RND heuristics.

The second scenario evaluates the unrestricted setting. We examine the received agent utility for the various heuristics with $M = 12$ and $k = 3$. For our results to generalise well to practical scenarios, M should be as large as possible. However, as we shall see later (see Figure 3), ES rapidly becomes computationally intractable as M increases; thus we set $M = 12$ as a compromise value that has a reasonable number of auctions but that is still tractable for ES. According to the ANOVA 2-factor test, all of our heuristics (DOM-KS, DOM-ES, EQT-KS,

⁶ Simulated annealing is applied as a global maximisation technique to identify the best bids. However, since it requires up to 1000 seconds to run even for this simplified setting, it is not a practical solution to our problem.

EQT-ES) are significantly better than the benchmark heuristics⁷. Amongst our heuristics, DOM-ES is significantly better than DOM-KS, whereas EQT-ES performs similarly to EQT-KS⁸. On average, our best strategies (EQT-KS, EQT-ES) improve the agent utility by 24% and 27% when compared with the GRD benchmark.

In more detail, the set of auctions can be characterised by the degree of overlap λ and the percentage that are sealed-bid (%SB). To assess how these factors affect our heuristics, the auction sets are classified into one of the four categories, C1 – C4⁹, in terms λ and %SB. The agent utility follows the order of EQT-ES \geq EQT-KS \geq DOM-KS \geq DOM-ES $>$ GRD $>$ RND and the ranking is the same for each of the four categories.

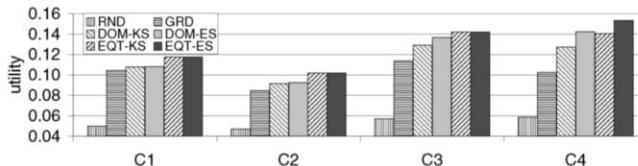


Figure 2. Agent utility for unrestricted auction setting.
($M = 12, k = 3, N = [5, 10]$ uniformly distributed)

In addition, our heuristics compare increasingly favourably¹⁰ to the GRD/RND benchmarks when $\lambda \geq 0.5$ and/or %SB < 0.5 . This is because many auctions are started or closed at about the same time if λ is large. Thus an agent can hunt for bargains by bidding in many auctions, while not taking too large a risk of accidentally purchasing too many items. When the percentage of English and Dutch auctions is high (i.e. when %SB is low), the agent receives more information in the bidding process. In short, both the careful management of excess procurement risk and improved utilisation of available information require much intelligence, and, thus, the more sophisticated heuristics are beneficial.

The DOM and EQT threshold heuristics require comparatively little computation because they adopt a simplified auction setting. However, the time taken to evaluate the more complex auction selection heuristics is an important practical consideration, especially when many decisions have to be made at the same time (which often occurs near the closing time of a set of simultaneous auctions). To this end, Figure 3 shows the average computation time for the heuristics when running on a modern PC. The ES heuristic is known to have exponential time complexity and it fails to keep up with practical requirements when M is larger than about 12. In contrast, the KS heuristic has been shown to remain acceptable for M up to at least 200 auctions with its pseudo-polynomial time complexity.

To summarise our results, the agent's utility is affected strongly by the threshold heuristic and EQT is the preferred

choice because it is better than DOM for each of the four categories of multiple auctions examined in the second scenario. For the auction selection heuristic, the computation time is an extra factor to consider in addition to the received agent utility. Here the ES heuristic can tolerate bad selection of bidding thresholds and its use should be considered if its computation time is deemed acceptable. But for a larger number of auctions ($M > 12$) the KS heuristic is more suitable because the sacrifice in performance when compared with ES is small (DOM-KS) or in some cases insignificant (EQT-KS).

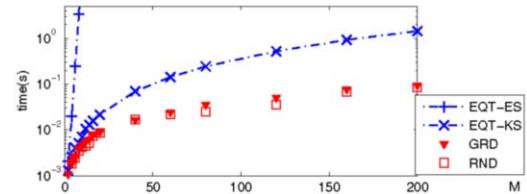


Figure 3. Average computation time for different heuristics.

4 Conclusions

This paper has developed and evaluated a novel two-stage heuristic as a bidding strategy for multiple heterogeneous auctions. In the first phase, a set of bidding thresholds is generated by the threshold heuristics, before a more sophisticated decision heuristic is applied to decide which subset of auctions to participate in. Our empirical evaluation shows that EQT is a better threshold heuristic than DOM and for the auction selection heuristic, KS is an acceptable choice especially for larger numbers of auctions. At present, the heuristics developed assume that there is only one global bidder in the market. However, as a future direction, we intend to examine the Nash equilibrium strategy and analyse the change in best response when more global agents operate in the system.

ACKNOWLEDGEMENTS

This research was undertaken as part of the EPSRC funded project on Market-Based Control (GR/T10664/01). This is a collaborative project involving the Universities of Birmingham, Liverpool and Southampton, BT and HP.

REFERENCES

- [1] P. Anthony and N.R. Jennings, 'Developing a bidding agent for multiple heterogeneous auctions', *ACM Trans on Internet Technology*, **3**, 185–217, (2003).
 - [2] C. Boutilier, M. Goldszmidt, and B. Sabata, 'Continuous value function approximation for sequential bidding policies', in *Proc. of UAI*, pp. 81–90, (1999).
 - [3] A. Byde, 'A dynamic programming model for algorithm design in simultaneous auctions', in *Electronic Commerce: Second International Workshop, WELCOM 2001*, volume LNCS 2232, pp. 152–163, (2001).
 - [4] A. Byde, C. Priest, and N. R. Jennings, 'Decision procedures for multiple auctions', in *AAMAS'02*, pp. 613–620, (2002).
 - [5] M. Dumas, L. Aldred, G. Governatori, and A.H.M. ter Hofstede, 'Probabilistic automated bidding in multiple auctions', *Journal of Electronic Commerce Research*, **5**, 25–49, (2005).
 - [6] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W.H. Freeman, 1979.
 - [7] V. Krishna, *Auction Theory*, Academic Press, 2002.
 - [8] V. Krishna and R.W. Rosenthal, 'Simultaneous auctions with synergies', *Games and Economic Behavior*, **17**, 1–31, (1996).
 - [9] P. Milgrom and R.J. Weber, *The Economic Theory of Auctions*, volume 2, chapter A Theory of Auctions and Competitive Bidding, II, 179–194, Edward Elgar, 2000.
 - [10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd edn., 1992.
- ⁷ Comparing our worst heuristic (DOM-KS: $\bar{u} = 0.113$) to the best benchmark heuristic (GRD: $\bar{u} = 0.101$), the null hypothesis that they perform the same is rejected at the 95% confidence level (CI) ($F = 46.6 >$ critical value $F_c = 3.86$).
- ⁸ The null hypothesis that DOM-ES ($\bar{u} = 0.120$) and DOM-KS ($\bar{u} = 0.113$) perform the same is rejected ($F = 9.28 > F_c = 3.86$). But, for EQT-ES ($\bar{u} = 0.129$) and DOM-KS ($\bar{u} = 0.125$), the null hypothesis that they perform the same is accepted at the 95% CI ($F = 3.32 < F_c = 3.86$).
- ⁹ C1: $\lambda < 0.5$ and %SB ≥ 0.5 , C2: $\lambda \geq 0.5$ and %SB ≥ 0.5 , C3: $\lambda < 0.5$ and %SB < 0.5 , C4: $\lambda \geq 0.5$ and %SB < 0.5 .
- ¹⁰ Our worst heuristic, DOM-KS, gains 3, 8, 14 and 24% higher utility than GRD for category C1-C4 respectively.

Are Parallel BDI Agents Really Better?

Huiliang Zhang and Shell Ying Huang¹

Abstract. The traditional BDI agent has 3 basic computational components that generate beliefs, generate intentions and execute intentions. They run in a sequential and cyclic manner. This may introduce several problems. Among them, the inability to watch the environment continuously in dynamic environments may be disastrous and makes an agent less rational – the agent may endanger itself. Two possible solutions are by parallelism and by controlling and managing the 3 components in suitable ways. We examine a parallel architecture with three parallel running components which are the belief manager, the intention generator and the intention executor. The agent built with this architecture will have the ability of performing several actions at once. To evaluate the parallel BDI agent, we compare the parallel agent against four versions of sequential agents where the 3 components of the BDI agent are controlled and managed in different ways and different time resources are allocated to them. Experiments are designed to simulate agents based on the sequential and parallel BDI architectures respectively and the ability of the agents to respond to the same sequences of external events of various priorities are assessed. The comparison results show that the parallel BDI agent has quicker response, react to emergencies immediately and its behaviour is more rational.

1. INTRODUCTION

As identified in the survey of agent architectures [15], three kinds of agent architectures, deliberative architecture [1, 13, 9], reactive architecture [2] and hybrid architecture [6], are proposed according to the processing mechanism of the agents. The traditional deliberative architecture has 3 basic computational components that generate beliefs, generate intentions and execute intentions. For an external event, an agent will process it by going through the belief generation, intention generation and intention execution in a sequential manner. For example, in PRS (procedural reasoning system) [9], the main control process runs in iterations. The agent will not proceed to the next step until the current step is finished. In a complicated and dynamic environment, the agent may need more time to search for proper intentions or actions need more time to be executed. Then the agent cannot perceive new information in time and the reactivity of the PRS agent cannot be assured.

A possible solution appears in the TouringMachine [6]. For each time cycle, the control framework has fixed time resource to use. This method ensures that the agent can sense the environment at fixed time intervals. The probability of overlooking emergencies

is reduced if the portions of the time spent in detection and processing are well balanced. But it is usually advised that the detection should not consume much time. So the problem of poor reactivity still exists.

In agents based on BDI logics, such as AgentSpeak(L) [12] and LORA (Logic of Rational Agents) [16], these problems may induce the slow reactivity and intention reconsideration issues. For example, in LORA, desires, intentions, and actions are generated based on belief. The original circumstance/belief may have changed during these processing. So the intentions may become impossible under new circumstance and the agent should not commit to the infeasible intentions. An improvement was made by updating belief and reconsidering intentions after executing each action. An experiments result of intention reconsideration by Kinny and Georgeff is provided in the book [16]. The result shows that the more frequently the intentions are reconsidered, the lower the effectiveness of the agent is.

Utilizing a more powerful computer is a method to lessen the problems. An alternative is to investigate what improvement can be achieved by controlling and managing the 3 basic computational components of a BDI agent in different ways. Another solution can appear in parallelizing the agents' basic actions. In [10], a multi-threaded approach is shown to have obvious advantages in lessening the impact of IO operations in the simulation of an intelligent agent like a human soccer player. From the view of a real human being, a person is always doing several things simultaneously. The brain engages many processes like planning, walking and watching for traffic continuously and at the same time. A parallel BDI agent architecture has been proposed in [17]. The architecture consists of the belief manager, the intention generator and the intention executor running in parallel. The activities of these components are coordinated by interrupts. The agent built with this architecture has the ability of reacting to urgent matters immediately and performing several actions at once.

In this paper we evaluate the parallel BDI agent by comparing it against four regular versions of sequential agents where the 3 components of the BDI agent are controlled and managed in various ways. Experiments are designed to simulate agents based on the sequential and parallel BDI architectures respectively and the way they respond to the same sequence of external events are evaluated. The comparison results show that the parallel BDI agent has quicker response and its behaviour is more rational.

The remainder of this paper is structured as follows. In section 2, we introduce the four sequential BDI agents each with their own way of controlling and managing their computational components.

¹School of Computer Engineering, Nanyang Technological University, Singapore 639798, email: {pg04043187, ASSYHUANG}@ntu.edu.sg

In section 3, the parallel BDI agent architecture proposed in [17] is summarized. The simulation experiments are described and the results are analyzed in section 4. Finally, we summarize the paper and propose possible applications of the parallel BDI architecture.

2. SEQUENTIAL BDI AGENTS

There are 3 basic computational components in a BDI agent, the belief manager to *detect* the changes in the environment to generate new beliefs, the intention generator to *deliberate* on the beliefs to generate new intentions and the intention executor to *execute* the intention plans. In a sequential agent, only one computational component is running at any time. We present several ways of controlling and managing the components in an attempt to get better performance from a sequential agent.

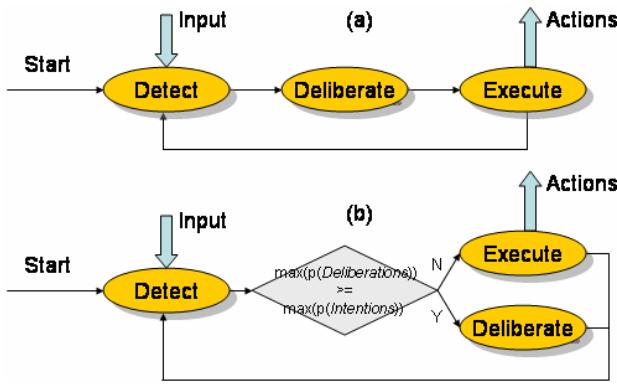


Figure 1. Sequential BDI agents.

- (1) As shown in Figure 1(a), the 3 components run in a cyclic way and each uses up the pre-allocated and fixed time resource. The deliberation/intention cannot be suspended and resumed. If the remaining time of a component (only the *deliberate* and the *execute* components) is not sufficient for a deliberation/intention to be finished, the remaining time will be wasted.
- (2) A more flexible way is to allocate time resources to the *deliberate* and *execute* components only when needed. If a component has nothing to do, it terminates and the next component starts. In order to keep the agent vigilant, the *detect* component always uses up all its allocated time. The actual time used for *deliberate/execute* should not exceed the maximum pre-allocated time to such a component. This agent has a cycle time ranging from a minimum that is the fixed time for the *detect* component to a maximum that is the sum of the allowable times of the 3 components.
- (3) This is a variant of the agent (2). It suspends a task when the time allocated to the current component is used up and resumes it when the component's turn comes in the next cycle. For example, the *execute* action can start an *intention* which costs 5 time units when there is only 1 time unit remaining.
- (4) This agent has a cycle as shown in Figure 1(b), in each cycle, after the *detect* component, the agent will choose to *deliberate* or *execute* based on the maximum priority of *deliberations* and *intentions*. After *deliberate/execute* is finished, another cycle begins. This makes the agent more watchful for emergencies.

The characteristics of the four sequential BDI agents are summarized in Table 1. In all the sequential agents, when there are

multiple deliberations/intentions to handle in the *deliberate/execute* component, the one with highest priority will be processed first. The performance of these agents will be compared with the parallel BDI agent.

Table 1. Sequential agents.

Agent no	Flexible time allocation?			suspend-resume?	Illustration
	detect	deliberate	execute		
1	N	N	N	N	Figure 1(a)
2	N	Y	Y	N	Figure 1(a)
3	N	Y	Y	Y	Figure 1(a)
4	N	Y	Y	Y	Figure 1(b)

3. PARALLEL BDI AGENT

The detailed architecture of the parallel BDI agent is proposed in [17]. Figure 2 shows an abstract representation of it. The salient features of the architecture are

- (a) The *belief manager*, the *intention generator* and the *intention executor* run continuously and in parallel. So the agent is capable of performing several actions at once. For example, the agent is able to perceive the environment, deliberate on its desire and belief to generate the next intention and execute the plan for the current intention all at the same time. This makes the agent more human-like.
- (b) The parallel running components communicate through interrupts. Interrupts have different priorities representing the importance and urgency of the desires/beliefs/intentions to be handled. After receiving an interrupt, the next action of the component will be decided by the priority of the interrupt. If the new information needs to be dealt with immediately, the component will suspend the current job and start to process the new situation. If the urgency of the new information is not higher than the current job, the new information will be kept for future processing at an appropriate time. The interrupt mechanism ensures that an emergency can be dealt with rapidly. Thus, the agent is capable of quick reaction to emergencies.

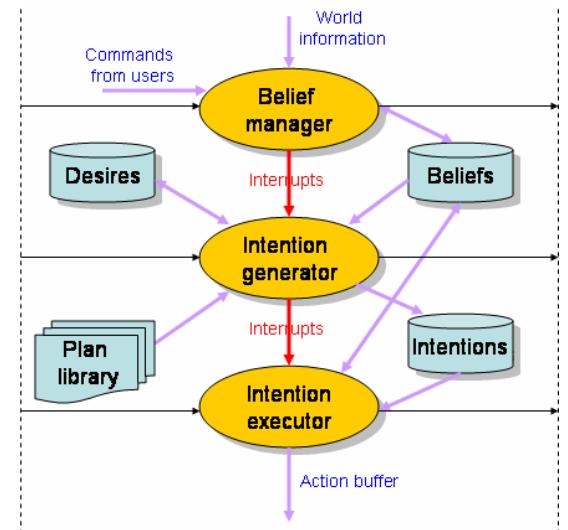


Figure 2. Parallel BDI model.

4. EXPERIMENT

4.1 Sequential BDI Agents Configuration

This is to allocate time resources to the 3 components of a sequential BDI agent. We used three time allocation schemes for sequential agents according to their emphasis on the three components. For a maximum cycle of 15 time units, three schemes showing the fixed or maximum allowable time quota for each component of the BDI agent are given in Table 2.

Table 2. Allocation schemes.

Configuration	detect	deliberate	execute
C1	1	4	10
C2	3	4	8
C3	5	3	7

The numbers in Table 2 are not completely arbitrary: (1) configuration C1 puts more emphasis on executing intentions with the risk of overlooking emergencies; (2) C3 gives more time to the *detect* component to be more vigilant, but the time for the *execute* component is cut; (3) C2 is a compromise between C1 and C3.

Each sequential agent described in Section 2 will be configured according to C1, C2 and C3 respectively in the experiments to compare them with the parallel agent.

4.2 The Input Data

The evaluation of the sequential and parallel agents is done by simulation of the processing of events by agents. All the sequential agents and the parallel agent will process some sequences of events. Each event will be processed by the 3 computational components of the BDI agent, namely, the belief manager(*detect*), the intention generator(*deliberate*) and the intention executor(*execute*).

To evaluate the agent ability to handle events of different importance or urgencies, events will have one of the four different priority levels 1 to 4, with 4 being the highest. We assume that an event can be detected and a belief generated in 1 time unit and each deliberation to generate an intention takes 2 time units. The intention execution time of events at all priority levels is uniformly distributed in the range from 1 to 7 time units. So the average intention execution time is 4 time units. This also means the average time required to handle an event is 7 (1+2+4) time units.

We use the exponential density function to represent the inter-arrival time between any two events. As shown in [14], the exponential density function is memoryless and often used to model lifetimes or waiting times. The cumulative distribution function is shown as:

$$cdf(t) = 1 - e^{-\lambda t} \quad (1)$$

where, $1/\lambda$ is the mean; t is the time units.

$cdf(t)$ shows the probability that the inter-arrival time between 2 events is less than t . So given a total number of events, *sum*, to be used in our experiments, we can decide the number of the intervals with length of n time units by:

$$G(n) = (cdf(n) - cdf(n-1)) * sum \quad (2)$$

If the *sum* of the events is provided, the numbers of the intervals with different lengths can be decided. Then the intervals are selected randomly for the events. The current time plus the interval

length is the arrival time of the next event. The event priority is selected randomly from 1 to 4.

We consider three sequences of events with different average inter-arrival times. The average inter-arrival times of the 3 sequences of events are respectively smaller than, equal to and larger than the average processing time required by an event. The events statistics used in the experiments are shown in Table 3.

Table 3. Events statistics.

Set	Expected average interval $1/\lambda$	Events count				Actual average interval	
		Priority			sum		
		1	2	3			
a	4	27	20	26	23	96	
b	7	24	22	24	25	95	
c	12	25	20	22	24	91	

When an event arrives and the agent is not doing detection, the event will be stored in an event buffer. The agent can retrieve the new events later. After receiving the event, the agent will create one *deliberation* for it. The *deliberation* will be added in a deliberation queue. The *deliberation* and *intention* plan selected later will have the same priority as the event.

4.3 Results and Analysis

We will use the *response time* to evaluate how well the agent processes the event. The *response time* is defined as the time between the arrival of the event and end of the execution of the intention plan chosen for the event. The response time is calculated as the sum of the time for detecting the event, deliberation and execution. The overhead transmit messages between the three parallel components in the parallel agent is omitted as it is felt that the delay in passing the interrupts is very small.

The results of the experiments are presented in Table 4. The ART is the Average Response Time of all events. ART_p stands for the ART of the events with priority p . ART_w is the weighted ART by the priorities of the events.

$$ART = \frac{\sum_{i=1}^{Count} T_i}{Count}, \quad ART_w = \sum_{p=1}^4 \frac{ART_p * p}{1+2+3+4} \quad (3)$$

Where, T_i is the response time for event i ; *Count* is the number of events.

Parallel agent vs sequential agents

Looking at the response times of the parallel agent (rows with agent no. 5) and the various sequential agents (rows with agent nos. 1 to 4), we can see that the parallel agent responds at least 3 times faster than the sequential agents in most the cases. This shows that the parallel agent can process events quicker than the sequential ones. The main reason for this difference comes from the fact that the parallel agent is able to use 3 CPUs while each sequential agent uses one. To give the sequential agents a single CPU which is 3 times as powerful as the CPU used in the parallel agent will enable the sequential agents respond faster but they will still not be able to guarantee immediate handling of high priority events.

One thing to note is that in a few cases the parallel agent (agent 5) responds less than 3 times faster than the sequential ones. This happens with sequential agents 2, 3, 4 with configurations C1

Table 4. ART of the events by the agents.

Con fig.	Set	Agent no	ART _p				ART	ART _w
			1	2	3	4		
C1	a	1	380.3	262.2	73.0	16.5	185.3	119.0
		2	225.0	170.1	40.3	15.6	113.4	74.9
		3	225.0	167.3	40.0	16.7	113.0	74.7
		4	226.4	168.3	41.0	17.7	114.1	75.7
		5	32.4	14.0	11.6	6.5	16.7	12.1
C1	b	1	292.0	32.3	21.4	16.3	90.9	48.6
		2	63.9	20.6	16.5	14.4	28.9	21.2
		3	55.5	25.3	18.3	15.2	28.5	22.2
		4	77.7	19.9	18.2	14.5	32.7	23.0
		5	14.1	8.9	8.2	6.5	9.4	8.2
C1	c	1	43.8	21.8	16.5	15.7	24.9	20.0
		2	16.3	10.2	9.8	9.1	11.5	10.3
		3	16.4	10.6	10.1	9.1	11.7	10.4
		4	19.3	11.8	11.3	10.2	13.3	11.8
		5	11.4	7.0	6.4	6.3	7.9	7.0
C2	a	1	426.4	296.2	104.0	17.4	214.0	140.0
		2	330.3	253.6	79.0	14.7	170.7	113.4
		3	318.4	239.4	62.5	18.9	160.9	106.0
		4	321.9	242.2	66.4	16.2	162.9	107.1
		5	32.4	14.0	11.6	6.5	16.7	12.1
C2	b	1	387.7	83.9	27.6	16.6	128.7	70.5
		2	266.4	40.9	21.3	16.7	86.6	47.9
		3	224.3	39.0	28.8	18.8	77.9	46.4
		4	265.5	31.9	23.4	16.8	84.8	46.7
		5	14.1	8.9	8.2	6.5	9.4	8.2
C2	c	1	70.6	29.7	18.2	15.6	34.4	24.7
		2	32.9	13.8	11.4	11.6	17.9	14.1
		3	25.7	14.4	13.7	14.9	17.5	15.5
		4	37.2	19.5	17.1	15.4	22.7	18.9
		5	11.4	7.0	6.4	6.3	7.9	7.0
C3	a	1	1016.5	666.2	359.8	22.9	527.6	351.9
		2	645.1	422.3	177.3	13.3	320.7	207.5
		3	471.8	350.4	139.3	19.4	248.1	166.8
		4	480.7	360.4	148.7	20.0	255.4	172.8
		5	32.4	14.0	11.6	6.5	16.7	12.1
C3	b	1	936.0	677.3	86.7	20.4	425.6	265.2
		2	633.9	269.4	26.6	14.0	232.9	130.8
		3	508.1	87.2	34.5	21.4	162.9	87.2
		4	523.2	121.4	32.0	18.8	173.3	93.7
		5	14.1	8.9	8.2	6.5	9.4	8.2
C3	c	1	799.9	218.5	35.1	18.9	281.3	141.8
		2	219.3	26.0	13.4	11.7	72.3	35.8
		3	55.4	30.0	16.7	16.1	30.1	23.0
		4	71.9	37.8	23.6	15.9	38.0	28.2
		5	11.4	7.0	6.4	6.3	7.9	7.0

and C2 when the event sequence is set c. The ARTs and the weighted ARTs of agents 2, 3, 4 are less than 2 times those of agent 5. This means when the inter-arrival time is long enough, the parallel agent will not show its advantage. Configuration C3 is allocating too much time to the *detect* component of the agent so the sequential agents responds more than 3 times slower than the parallel agent except for the priority 4 events.

The priority 4 events are the highest priority events in the experiments. The average time needed to process one such event in the ideal case (ATN_4) is calculated by:

$$ATN_4 = \frac{\sum_{i=1}^{\text{Count}} (TDetection_i + TDeliberation_i + TExecution_i)}{\text{Count}} \quad (4)$$

where, $TDetection_i$ is the time used to detect the event i ; $TDeliberation_i$ is the time used to deliberate the event i ; $TExecution_i$ is the time used to execute the plan generated based on the event i ; $count$ is the number of events with priority 4.

In this experiment, $TDetection$ equals to 1 and $TDeliberation$ equals to 2. $TExecution$ can be obtained from the events list. Using the equation, ATN_4 equals to 6.26 for set a, 6.48 for set b, 6.3 for set c. Compared to the ART_4 (column 7 in Table 4) we can see that in the parallel agent, the events with the priority 4 from set b and c

are processed immediately. It spends just 0.24 time units more for set a. Set a is a sequence of events with an inter-arrival time smaller than the processing time required for an event. So the intention generator and the intention executor are busy handling other beliefs and intentions when a high priority belief/intention comes to them. Here the interrupt mechanism in the parallel agent is able to guarantee immediate handling of higher priority items while the sequential agents are not able to do this. It is clear that the parallel agent has a big advantage over the others on processing real emergencies.

For events with lower priority, the difference of ART between the sequential agents and the parallel agent is bigger.

With different time resources allocation in sequential agents

Looking at the ART and ART_w columns and comparing the corresponding rows for configurations C1, C2 and C3, we conclude that the performances of the sequential agents with configuration C3 are significantly worse than those with configuration C1 and C2. This shows that the sequential agents perform badly if they spend more time on detecting and less time on deliberation and intention execution. The processing of emergencies is often postponed, though the emergencies are detected earlier in configuration C3. This can be seen that in most cases (event sets a, b, c and agent number 1 to 4) the processing of the highest priority events also have longer response time. This indicates that in real life, the agent is not reacting to high priority events quickly and is taking a longer time to react to other events.

We also observe that the performances of the sequential agents with configuration C1 is significantly better than C2. This shows that the sequential agents perform much better if they spend shorter time on detecting and more time on deliberation and intention execution. Because the deliberating and executing components get more time resources, the beliefs and intention plans get cleared faster so the events experience shorter response time.

With different ways of controlling the computational components in sequential agent

Looking at the ART and ART_w columns and comparing the corresponding rows among agent no. 1, 2, 3 and 4, we see that agent 1 is the loser in all cases. This is expected because of its rigid way of controlling the *detect*, *deliberate* and *execute* components. In the best performing configuration C1, agents 2, 3 and 4 have comparable performance in all event sequences a, b and c. In configurations C2 and C3, generally agents 2 and 3 are better with 3 being the best more often but agent 4 is not the loser every time. So we conclude that it is better to allow the computational components to start processing an item if there is one, then suspend it when its time quota is up and resume in the next round. This works together with the policy that if a component has nothing to do, it will give way to the next computational component.

5. CONCLUSION

In this paper, five ways of organizing and controlling a BDI agent are studied, including the sequential agents and the parallel one. They are evaluated by simulating their operations in processing events of different priorities and examining their performance. We analyze their performance in handling the same sequences of

events. The results show that the parallel BDI agent outperforms the sequential ones in offering significantly shorter response time to events with various inter-arrival times. The parallel BDI agent with its interrupt mechanism is able to guarantee to immediately react to high priority events where none of the sequential ones are capable of. The advantage comes from the following facts:

- The agent process detection, deliberation and execution concurrently. So these actions are carried on at the same time.
- The events can be detected immediately. This ensures that the emergencies will not be overlooked.
- The *deliberation/intention* with higher priority can be performed first by preempting others.

This architecture is useful in implementing agents working in complex and dynamic environments. In the following part, two possible utilities and one problem of the parallel BDI agent architecture are explained.

This architecture should be helpful in the research of continual planning. Continual planning means that the agent will be continuously planning, interleaving planning with execution, because its plans can undergo continual evaluation and revision [3]. For a parallel agent, it will be able to continue with planning while executing an intention plan. Several techniques have been produced for continual planning. For example, the CPEF (continuous planning and execution framework) [11] integrates HTN planning technique [4, 5] to implement open-ended reasoning. Open-ended planning process allows the agent not to generate full level plans before execution. Obviously, the techniques for continual planning increase the needs for time resource. In order to replan, it is better for the agent to detect new situations frequently. So the parallel architecture is ideal to support continual planning.

Another possible utility of the architecture is that it can provide the agent some adaptive behaviour. Adaptive ability is an important attribute for agents to show the autonomy and proactiveness properties [7]. An experimental step was taken in [8]. The learning is implemented by applying the Top-down induction of decision trees on the agent's action models. The models are labeled with success or fail tag. In the situations with fixed action steps, the agent can interact with environment with past experiences. But for agents working in continuous environment, the limitation is obvious: the models may be too voluminous to save. It is possible to extend the original BDI model by adding an experience function library. Some computationally expensive algorithms can be coded in this library and called. The parallel BDI agent will be able to spend time deliberating to learn from past experiences while perceiving the environment and executing intentions at the same time.

As for the sequential agents, we found that it is better to allow more time resources for the deliberating and executing components of the BDI agent. Furthermore, the deliberating and executing components should fully utilize their time quota by suspending unfinished work at the end of the time slice and resuming it next round. When there is nothing to do, it will give way to the next computational component.

REFERENCES

- [1] M. E. Bratman, *Intentions, Plans, and Practical Reason*, Harvard University Press, Cambridge, MA, 1987.
- [2] R. A. Brooks, *Elephants Don't Play Chess. Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, The MIT press, London, 3-15, 1991.
- [3] M. E. DesJardins, E. H. Durfee, Jr. Charles, L. Ortiz, and M. J. Wolverton, 'A survey of research in distributed, continual planning', *AI Magazine*, 21(4), Winter 2000.
- [4] K. Erol, J. Hendler, and D. S. Nau, HTN planning: Complexity and expressivity, *Proceedings of the Twelfth National Conf. on Artificial Intelligence*. AAAI Press/MIT Press, 1123-1128, 1994.
- [5] K. Erol, J. Hendler, and D. S. Nau, Semantics for Hierarchical Task-Network Planning, Technical report CS-TR-3239, UMIACS-TR-94-31, ISR-TR-95-9 at the University of Maryland, 1994.
- [6] I. A. Ferguson, TouringMachines: An architecture for dynamic, rational, mobile agents, Ph.D. thesis, Comput. Sci. Lab., Univ. Cambridge, U.K., 1992.
- [7] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, The belief-desire-intention model of agency, *proceedings of 5th Int. Workshop Intelligent Agents*, J. Muller, M. P. Singh, and A. S. Rao, Eds., Heidelberg, Germany, 1-10, 1999.
- [8] A. Guerra-Hernandez, A. El Falla-Seghrouchni, and H. Soldano, Learning in BDI Multi-agent Systems, *Proceedings of CLIMA IV- Computational Logic in Multi-Agent Systems*, Fort Lauderdale, FL, USA, 218-233, January 6-7, 2004.
- [9] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, 'An architecture for real-time reasoning and system control', *IEEE Expert*, 7(6):34-44, 1992.
- [10] K. Kostiadis, and H. S. Hu, A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition, *Proceedings of the 3rd Int. Workshop on RoboCup (IJCAI-99)*, 1999.
- [11] K. L. Myers, 'CPEF: A continuous planning and execution framework', *AI Magazine*, 20:63-70, 1999.
- [12] A. S. Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away*, Springer-Verlag New York, Inc. Secaucus, NJ, USA, 42-55, 1996.
- [13] A. S. Rao, and M. P. Georgeff, BDI agents: from theory to practice, *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, 1995.
- [14] J. A. Rice, *Mathematical Statistics and Data Analysis*. Duxbury Press, 1999.
- [15] M. J. Wooldridge, and N. R. Jennings, 'Agent Theories, Architectures and Languages: A Survey', In M.J. Wooldridge and N.R. Jennings (Eds.): Intelligent Agents. Lecture Notes in Artificial Intelligence, Springer Verlag, Vol. 890, 1995.
- [16] M. J. Wooldridge, *Reasoning about rational agents*, The MIT Press, Cambridge, Massachusetts, London, England, 2000.
- [17] H. L. Zhang, and S. Y. Huang, A Parallel BDI Agent Architecture, *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2005)*.

Dynamic Control of Intention Priorities of Human-like Agents

Huilang Zhang and Shell Ying Huang¹

Abstract. Intention scheduling mechanism plays a critical role in the correct behaviour of BDI agents. For human-like agents, the independent intentions should be scheduled based on their priorities, which show the respective importance and urgency of the intentions. We propose to enrich the BDI agent architecture with 2 processing components, a PCF (Priority Changing Function) Selector and a Priority Controller. These enable a BDI agent to assign an initial priority value to an intention and change it with time according to the chosen PCF. The initial priority value reflects its urgency at the time of intention creation. The PCF selected defines how the priority should change with time. As an example, we design a function by simulating human behaviors when dealing with several things at the same time. The priority first increases with time according to the Gaussian function to simulate that people are more inclined to do something which has been on their mind for sometime. After a certain time, if the intention still does not get executed because of other higher priority intentions, its priority will decrease according to the Ebbinghaus forgetting curve. Experiment results show that with this mechanism, the agent can show some human-like characteristics when scheduling intention to execute. This can be used when simulating human-like agents.

1. INTRODUCTION

BDI (belief-desire-intention) model is the most famous one of the deliberative agent architectures [2, 10]. Several successful agent architectures and systems based on BDI have been developed. PRS (Procedural Reasoning System) is an implementation of the BDI model. In each cycle, the belief is updated first. Then intentions are selected from the applicable plans. Finally, actions in the chosen intention plan are executed [6, 7]. AgentSpeak(L) [11] and LORA (logic of rational agents) [13] are two sets of operational semantics defined for BDI agents. The decision is made through logic reasoning.

However, the scheduling of intention execution is usually omitted in these BDI systems. The researchers mostly concentrate on solving the problem of intention generation. For example, in AgentSpeak(L) [11], the selection function S_I selects an intention to execute from the intention set I . The detailed selection criteria are not specified. We believe the scheduling of intention is crucial in an agent's ability to cope with the changing world. Some scheduling mechanisms appear in subsequent researches. In AgentSpeak(XL) [1], an extension to the AgentSpeak(L), a task

scheduler is incorporated into the interpreter to decide how to select intentions. The set of intentions in the AgentSpeak(L) is converted into a corresponding TÆMS task structure. Then the selection is done based on the analyses of the relationship among the plans in the TÆMS task structure. The 'enables' and 'hinders' relationships indicate which plan may be executed first. Another method is shown in the JAM agent architecture [5]. The intention selection is done based on the utility value of the plan. The intention with higher utility will be executed first. Recently, another work of intention scheduling is reported in [9]. The researchers take several properties into consideration when scheduling the intentions, such as the importance of the plan, the estimated running time, the deadline utility function, the degree of completeness and FairFacter. In a motivation-based planning and execution framework[3], it was proposed to assign goals and actions priority values indicating their importance based on the agent's motivations. In their work, the strength of the agent's motivations changes with the changes in the environment.

We consider the problem of intention scheduling in an agent who will behave like an "average human". If people identify and accept an agent as human and not machine-like, they tend to trust the agent better. For example, a companion to shut-ins or a playmate for a child should display a human way of handling intentions. We use a single priority value to decide the scheduling order of independent intentions like intentions to eat, to sleep, to play with a friend, to play a certain game or to do school homework. The priority represents the importance and urgency of the intentions to an agent. The priority value is expected to change with time. Currently, the control of the priority changing with time has not been adequately researched even though some work has been done in the artificial life community. In [8], a priority control mechanism for behavioural animation is proposed. The priority is set at minimal value immediately after the agent display a certain behaviour like drinking. Then this priority is increased with time. The increased priority will induce the agent to drink again. However, expecting the priorities of all intentions to change in the same manner is not realistic. Different intentions should be allowed to change their priorities in various suitable ways. Some intentions may also change priorities with the arrival of new beliefs.

In this paper, we proposed an extended architecture to the original BDI architecture in order to support the capabilities of giving different intentions different priorities and changing them dynamically. *Priority Changing Functions* are pre-defined and are associated with the corresponding intentions. A Priority Controller

¹ School of Computer Engineering, Nanyang Technological University, Singapore 639798, email: {pg04043187, ASSYHUANG}@ntu.edu.sg

will change the priority value of the intentions along the time clock. As an example, we design a function which simulates the human behaviours when dealing with several things at the same time. It first increases the priority value according to the Gaussian function and then decreases according to the Ebbinghaus forgetting curve. We call this *reminding-forgetting* function. With the setting of suitable parameter values, this function is also able to simulate the changing of intention priority when a person is not very motivated to put an intention into actions. The function can also represent the changing of intention priority when an intention will get stronger and stronger and stay at its maximum value until it is carried out. To simplify the work, only independent intentions are considered.

The remainder of this paper is structured as follows. In section 2, we present the BDI agent architecture incorporated with an extension of priority control. In section 3, a human-like *priority changing function* (*reminding-forgetting* function) is discussed. And the experiment results are given out in section 4. At the end of the paper, we summarize the contribution of the mechanism and prospects of the future applications.

2. AGENT ARCHITECTURE

A typical BDI system is shown in Figure 1. The agent perceives the environment through the detection system. This information input will be organized into the *beliefs* structure. Then the *deliberate* action begins. Based on the current beliefs, the *desires* are updated and *intentions* generated. Finally, an intention from the *intentions* structure is chosen to be executed. In AgentSpeak(L) language, this is denoted as: $S_i(I) = i$. For independent intentions, they will be weighted by their importance or urgency. This is represented by the priority value. Higher importance, higher priority value will be assigned.

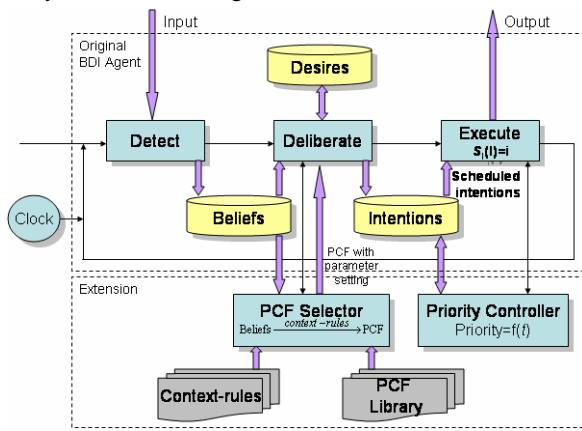


Figure 1. The agent architecture.

As shown in Figure 1, we introduce 2 processing components to the original BDI agent, a PCF (Priority Changing Function) Selector and a Priority Controller. When an intention is generated, the PCF Selector will, based on some context-rules, (i) select a suitable PCF for the new intention from the PCF Library and (ii) decide on the values of the parameters if any for the PCF. The Priority Controller will be responsible for updating the priorities of the intentions according to its PCF as time passes. Each time the BDI agent is to select an intention to execute, the Priority Controller will update the priorities of intentions by:

$$\text{Priority} = f(t) \quad (1)$$

Where, $f(t)$ is the PCF and t is the time.

As an example, for an intention to be completed before a deadline, a simple PCF is:

$$f(t) = \begin{cases} \alpha + \beta * (t - \text{start}) & \text{start} \leq t \leq \text{deadline} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Where, start is the time when the intention is generated, α is the initial priority value, β is the rate of increase of the priority value.

The PCF Selector is to choose the function and decide on the values of α and β based on how close start is to the deadline. The Priority Controller will compute the value of $f(t)$ as time passes.

Various PCFs suited to different intentions can be pre-defined. This enriches the BDI agent with the ability of realizing the scheduling of the intentions in a more realistic way. In the following section, we will show a PCF which is designed by simulating human behaviours.

3. A HUMAN-LIKE PCF

The basic PCF is shown as:

$$f(t) = \begin{cases} \text{Maximum priority} * I(t - \text{start}) & \text{start} \leq t \leq \text{deadline} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where, *Maximum priority* is the highest value which an intention can achieve; $I(t)$ is the function of influence factor with a range [0, 1].

Normally, humans tend to pay more interests to a new task. After a while, the interests will be reduced and sometimes the task is forgotten. There are also situations where someone has the intention to do something but he is either too lazy to do it or the interest is just not enough, so from the beginning the intention gradually fades away. For some other intentions, the priority will increase till it reaches its maximum and never fade away till it is executed. To simulate these behaviours, $I(t)$ is divided into two modes, reminding and forgetting.

1. Reminding

As time passes, a person will be more inclined to execute a pending intention. The priority of the intention should be increased gradually until it reaches a maximum value. We call this phase ‘reminding’. When the priority value is increased, the intention will have a higher chance to be executed than before. For this phase, we believe that: (1) the value of $I(t)$ should be increased gradually; (2) the gradient of $I(t)$ should be decreasing. This means the priority of the intention will be increased gradually and the rate of increase is decreased gradually. At the initial period, the priority will increase quickly. After that, the priority is increased in a smaller pace and the agent prepares for next phase, forgetting the intention. This so-called *reminding* process will continue until the priority reaches its maximum value ($I(t)$ reaches 1). We use the Gaussian function to simulate the influence factor in this process.

$$G(x) = \eta * \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right) \quad (4)$$

where, η is a constant value;

σ is the width of Gaussian function;

x_0 is the middle point of the function.

In order to make $I(t)$ reach maximum value 1 at the middle point, we set η to $\sqrt{2\pi}\sigma$. So the function to simulate the reminding process is shown as:

$$I(t) = \exp\left(-\frac{(t-t_m)^2}{2\sigma^2}\right) \quad (5)$$

where, t_m is the time where an intention starts to fade in memory; t is the elapsed time since the intention is created.

Simple differentiation will show that (1) the *reminding* function is increasing; (2) the rate of increase of the *reminding* function value first increases and then decreases before it reaches the middle point t_m .

2. Forgetting

If an intention is deferred for a long time because intentions with higher priority keep on coming, the best way to deal with it is to discard it. This is similar to that humans forget to do something when they are doing something else more important. In biological science, this is a protective mechanism to ensure that human can learn new things. We tend to forget things that the external environment does not remind us of. So as time passes, the priority of the intention in an agent is decreased. If the priority decreases to a value below a threshold, the intention will be removed (forgotten). The first significant study on memory was performed by Hermann Ebbinghaus and published in 1885 as *On Memory*. Ebbinghaus was the first to describe the shape of the forgetting curve [4]. This curve is the biological base on which we simulate the process of intention retention. In [4], the forgetting curve is described as:

$$R = e^{-t/S} \quad (6)$$

where, R is the retention, which means the ability to retain things in memory; t is the elapsed time; S is the strength of memory, which means the durability of things in memory.

Thus, we can make $I(t)$ by combining the functions of the two processes together. The function is shown as:

$$I(t) = \begin{cases} \exp\left(-\frac{(t-t_m)^2}{2\sigma^2}\right) & \text{if } t \leq t_m \\ \exp\left(-\frac{t-t_m}{S}\right) & \text{otherwise} \end{cases} \quad (7)$$

It is noticed that if t_m equals to 0, the reminding process will be skipped. This happens when someone has the intention to do something but he is either too lazy to do it or the interest is just not enough, he(the agent) starts to forget about it according to the forgetting curve from the beginning. If S is set to infinity, the priority will increase according to the reminding function till it reaches its maximum and stay at its maximum so there is no forgetting.

At following, we will prove that the influence function $I(t)$ is continuous.

Proof:

When $t \neq t_m$, it is obvious that $I(t)$ is continuous.

Then we need to prove the continuousness of the function at the point t_m : $\lim_{\Delta t \rightarrow 0} [I(t_m + \Delta t) - I(t_m)] = 0$.

We can see that at the point t_m , $I(t)=1$.

If $\Delta t < 0$, because the normal Gaussian function is continuous, we have $\lim_{\Delta t \rightarrow 0} [I(t_m + \Delta t) - I(t_m)] = 0$.

$$\text{If } \Delta t > 0, \text{ we have } \lim_{\Delta t \rightarrow 0} \left[\exp\left(-\frac{t_m + \Delta t - t_m}{S}\right) - I(t_m) \right] = 0.$$

So $I(t)$ is continuous at t_m .

Thus, the influence function $I(t)$ is continuous at every point. The influence factor does not change significantly at any time.

4. EXPERIMENT

In this part, we will first show how the PCF designed in section 3 is implemented in an agent. Then the function is applied by a BDI agent. The performance of the agent is analyzed.

4.1 Implementation

When the reminding-forgetting function is associated with the intention, the following initial parameters should be decided:

Table 1. Intention parameters related to priority control

Name	Type	Explanation
BP (Basic Priority)	Float	It is decided by initial urgency of the intention.
MP (Max Priority)	Float	The maximum priority for this intention.
t_m	Integer	The time when the forgetting process begins.
S	Integer	Strength of memory. It is assumed that an intention with higher basic priority will have longer retention.
Threshold	Float	In forgetting progress, if the priority is below the threshold, the intention will be removed.

Using the given values, we can calculate the width of Gaussian function σ by:

$$\sigma = \frac{\sqrt{2}}{2} * t_m * \sqrt{\frac{-1}{\ln I(0)}} \quad (8)$$

where, $I(0) = \frac{BP}{MP}$. So at the time of creation, an intention starts at its basic priority.

In Figure 2, the change of priority for 4 sample intentions with basic priority 1, 2, 3 and 4, are shown assuming they are generated at the same time. The maximum priority is set to 1.5 times of the basic priority. Strength is set to 10 times of the basic priority. And t_m is set to 20 time units. The lines 1-4 correspond to the intentions

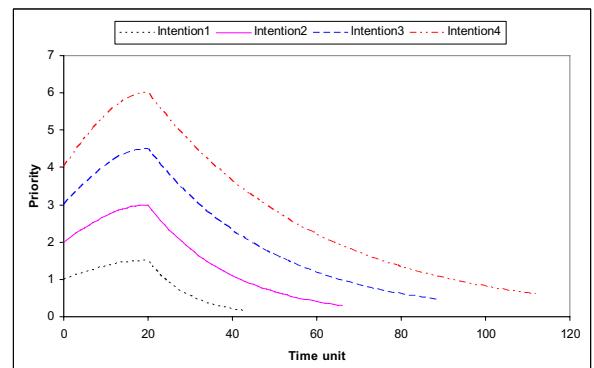


Figure 2. Priority Control of Four Intentions.

with basic priority 1-4. When their priority level is lower than threshold, set as 10% of the maximum priority, the intention will be removed (forgotten). The basic priority will affect the width of the Gaussian function and the strength of memory. With a higher basic priority, the intention will have a longer retention in memory. Another phenomenon is that for intentions created at the same time, the intention with higher basic priority will always has a higher priority level. This ensures that the intentions with higher urgency will be executed earlier.

4.2 Agent performance

In the following experiment, the agent will process a sequence of events. The events are created with random initial priorities from 1 to 4. We will first show how an event sequence is created. Then we will analyze how the agent processes the intentions with or without the *reminding-forgetting* PCF applied.

To evaluate the agent ability to handle events of different importance or urgencies, events will have one of the four different priority levels 1 to 4, with 4 being the highest. We assume that the intention execution time of events at all priority levels is uniformly distributed in the range from 1 to 7 time units. So the average intention execution time is 4 time units.

We use the exponential density function to represent the inter-arrival time between any two events. As shown in [12], the exponential density function is memoryless and often used to model lifetimes or waiting times. The cumulative distribution function is shown as:

$$cdf(t) = 1 - e^{-\lambda t} \quad (9)$$

where, $1/\lambda$ is the mean;

t is the time units.

$cdf(t)$ shows the probability that the inter-arrival time between 2 events is less than t . So given a total number of events, sum , to be used in our experiments, we can decide the number of the intervals with length of n time units by:

$$G(n) = (cdf(n) - cdf(n-1)) * sum \quad (10)$$

The intervals are kept in a vector. Then the intervals are selected randomly. The current time plus the interval length is the arrival time of the next event. The event priority is selected randomly from 1 to 4.

We consider three sequences of events with different average inter-arrival times. The average inter-arrival times of the 3 sequences of events are respectively smaller than, equal to and larger than, the average processing time required by an event. The events statistics used in the experiments are shown in Table 2.

Table 2. Events statistics.

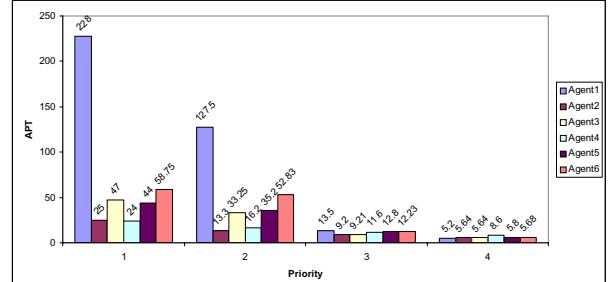
Set	Expected average interval $1/\lambda$	Events count				Actual average interval	
		Priority					
		1	2	3	4		
a	2	25	22	26	25	98	2.42
b	4	25	26	24	21	96	4.04
c	6	26	22	24	23	95	5.64

We experiment with 6 agents: agent 1 has a constant PCF so the initial priority is the priority all the time; agent 2 to 6 each has a different parameter settings for the PCF as shown in Table 3. The initial priority value of an intention is used as the basic priority.

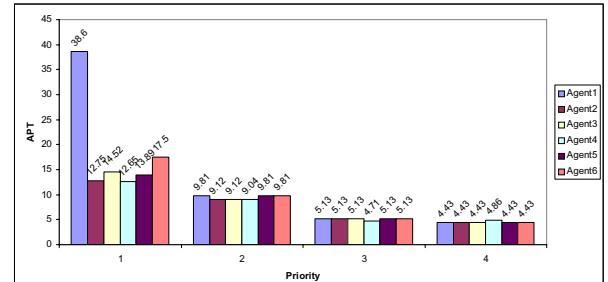
Table 3. Agents types.

Agent no	PCF	PCF parameter settings			
		S	T _m	MP	Threshold
1	-	-	-	-	-
2	✓	10*BP	20	1.5*BP	10%*MP
3	✓	10*BP	20	1.5*BP	1%*MP
4	✓	10*BP	20	2.5*BP	10%*MP
5	✓	10*BP	40	1.5*BP	10%*MP
6	✓	30*BP	20	1.5*BP	10%*MP

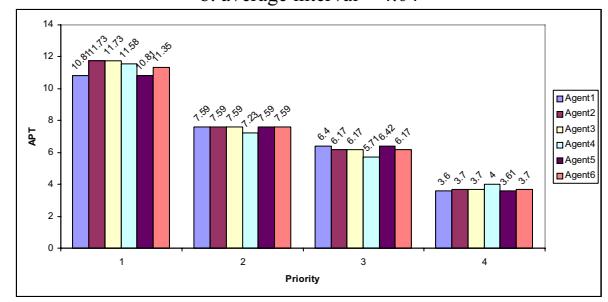
The intention processing time is defined as the duration from the time when the intention is created to the time when the execution of the intention is finished. The average processing time (APT) of the three sets of events by the agents is shown in Figure 3. In set *a*, compared to the expected average processing time 4, agent 1 does not have sufficient time to finish processing an event before the next event arrives. So the intentions with lower priorities have to wait for a long time. The time to process the events with priority 1 is 228 time units. So the intention is starved for a very long time. With the increased average inter-arrival time, the agent 1 has more time to process an event before next event begins and the corresponding APT is decreased to 58.6 for set *b* and 10.8 for set *c*. In agents 2 to 6, the APTs are the APTs of the events that get processed and they are smaller than those for agent 1. Here some intentions with priority 1, 2 or 3 in set *a* and *b* are forgotten due to a long waiting time, which can be seen from the statistics in Table 4. For events with priority 4, the APT is not



a. average interval = 2.42



b. average interval = 4.04



c. average interval = 5.64

Figure 3. APT of events.

affected too much, because the urgent events will be scheduled first. What we see is that many of those low priority events that experience terribly long waiting time in agent 1 are forgotten in agent 2 to 6 for event set *a*.

Looking at Table 4, we see the effect of the parameters of the PCF. For set *a*, 39 events are forgotten by agent 2 (row 2a). This is because the agent has no enough time to process all the crowded events. Rows 2b and 2c have most events processed because of the longer event inter-arrival time. Comparing row 2a and 3a, 5 more intentions are processed because of a lower threshold of retention. Comparing row 2a and 4a, 3 more intentions are processed because of a higher maximum priority. Comparing row 2a and 5a, 6 more intentions are processed because of a longer reminding period before forgetting starts. Comparing row 2a and 6a, 8 more intentions are processed because of a bigger strength of memory. The APT of them are a little bigger for intentions with lower priority in set *a* and *b*.

Table 4. Events processed statistics.

Agent no	Set	completed				sum	
		forgotten					
		Priority					
2	a	1 24	9 13	24 2	25 0	59 39	
	b	20 5	25 1	24 0	21 0	90 6	
	c	26 0	22 0	24 0	23 0	95 0	
3	a	3 22	12 10	24 2	25 0	64 34	
4	a	2 23	11 11	24 2	25 0	62 36	
5	a	3 22	11 11	26 0	25 0	65 33	
6	a	4 21	12 10	26 0	25 0	67 31	

5. CONCLUSION

In this paper, we propose to enrich the BDI agent architecture with 2 processing components, a PCF (Priority Changing Function) Selector and a Priority Controller. The priorities of the intentions can have different initial values and can be changed along the time according to the chosen PCF. The intentions can be scheduled according to their own nature. This provides us a chance to control the scheduling of the intentions in a more human-like manner.

As an example, a *reminding-forgetting* PCF is designed that simulates the way that human deals with several intentions together. The PCF goes through a rising/reminding phase and then a descending/forgetting phase. We proposed to use the Gaussian function to simulate the reminding processes of the intentions. And a forgetting function based on Ebbinghaus forgetting curve is used to simulate the function in forgetting process. With the setting of different parameter values, this PCF may also simulate a priority function that has the forgetting phase alone or the reminding phase alone. From the experiment results, we can see the difference in intention scheduling behaviour brought about by the PCF and the effect of setting the different parameter values.

The so called *reminding-forgetting* PCF is suitable for designing human-like agents. In the comparison experiment, the initial priority of all the intentions are assigned with the arbitrary settings of 1 to 4. To get a precise model of the intention scheduling process of humans, work needs to be done to adjust the parameter settings according to real human behaviours, such as the strength of memory, threshold, and so on.

REFERENCES

- [1] R. H. Bordini *et al.*, AgentSpeak(XL): efficient intention selection in BDI agents via decision-theoretic task scheduling, *proceedings of International Conference on Autonomous Agents*, 1294-1302, 2002.
- [2] M. E. Bratman, *Intentions, Plans, and Practical Reason*, Harvard University Press, Cambridge, MA, 1987.
- [3] A. M. Coddington, M. Luck, 'A Motivation Based Planning and Execution Framework', *International Journal on Artificial Intelligence Tools*, 13(1), 5-25, 2004.
- [4] Forgetting curve - Wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/wiki/Strength_of_memory.
- [5] M. J. Huber, JAM: A BDI-theoretic mobile agent architecture, *proceedings of third International Conference on Autonomous Agents (Agents '99)*, 236-243, 1999.
- [6] F. F. Ingrand, M. P. Georgeff, A. S. Rao, 'An architecture for real-time reasoning and system control', *IEEE Expert*, 7(6), 34-44, 1992.
- [7] J. Lee, M. J. Huber, E. H. Durfee, P. G. Kenny, UM-PRS: an implementation of the procedural reasoning system for multirobot applications, *proceesings of the AIAA/NASA Conference on Int. Robots in Field, Factory, Service and Space*, American Institute of Aeronautics and Astronautics, 1994.
- [8] F. Lamarche, S. Donikian, The Orchestration of Behaviours using Resources and Priority Levels, *Proceedigns of Computer Animation and Simulation 2001*, M.P. Cani, N. Magnenat-Thalmann, D. Thalmann (eds.), Manchester, UK, 171-182, September 2001.
- [9] Z.N. Lin, H.J. Hsu, F. J. Wang, Intention Scheduling for BDI Agent Systems, *Proceedings of International Conference on Information Technology Coding and Computing (ITCC2005)*. Volume 2, 2005.
- [10] A. S. Rao, M. P. Georgeff, BDI agents: from theory to practice, *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, 1995.
- [11] A. S. Rao, AgentSpeak(L): BDI agents speak out in a logical computable language, *proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world: agents breaking away*, 42-55, 1996.
- [12] J. A. Rice, *Mathematical Statistics and Data Analysis*, Duxbury Press, 1999.
- [13] M. J. Wooldridge, *Reasoning about rational agents*, The MIT Press, Cambridge, Massachusetts, London, England, 2000.

4. Knowledge Representation and Reasoning

This page intentionally left blank

Knowing Minimum/Maximum n Formulae

Thomas Ågotnes¹ and Natasha Alechina²

Abstract. We introduce a logical language with nullary operators $\min(n)$, for each non-negative integer n , which mean ‘the reasoner has at least n different beliefs’. The resulting language allows us to express interesting properties of non-monotonic and resource-bounded reasoners. Other operators, such as ‘the reasoner has at most n different beliefs’ and the operator introduced in [1, 4]: ‘the reasoner knows at most the formulae ϕ_1, \dots, ϕ_n ’, are definable using $\min(n)$. We introduce several syntactic epistemic logics with $\min(n)$ operators, and prove completeness and decidability results for those logics.

1 Introduction

In this paper we propose a logical language which allows us to say that a reasoner has at least n different beliefs (or knows at least n different formulae). We take a syntactic approach to epistemic and doxastic logics, which allows an agent’s beliefs to be, e.g., not closed under logical consequence, finite, and/or inconsistent. We extend the traditional language of belief logics, propositional logic with an additional unary operator B where $B\phi$ stands for ‘the agent believes ϕ ’, with nullary operators $\min(n)$, for every non-negative integer n . Such operators allow us to formulate interesting properties of reasoners, and the resulting logics have nice formal properties. We introduce several doxastic logics with $\min(n)$ operators, and show that they have (weakly) complete axiomatisations, and are decidable.

In the language with $\min(n)$, we can express dual operators $\max(n)$ (meaning: the reasoner has at most n different beliefs), and those operators, in turn, enable us to completely axiomatically describe reasoners with a bound n on the maximal size of their belief set. The bound on the number of distinct beliefs an agent can have, naturally corresponds to a bound on the size of the agent’s memory (assuming that each formula is a word of fixed size). Logics for agents with bounded memory were studied, for example, in [9], and more recently in [5]. However, in the language of standard epistemic logic it is impossible to express properties such as ‘the agent can apply the rule of modus ponens to its beliefs unless its memory is full’. We show later in the paper some examples of properties of bounded memory reasoners which become expressible in the language with $\min(n)$ operators.

Somewhat surprisingly, we can also define the ∇ operator introduced in [1, 4], where $\nabla\{\phi_1, \dots, \phi_n\}$ stands for ‘the agent believes at most the formulae ϕ_1, \dots, ϕ_n ’. The latter operator makes other useful properties easy to express. For example, formalising non-monotonic arguments becomes easier, because we can say in the language ‘the agent knows a set of formulae X and nothing else’.

The rest of the paper is organised as follows. In section 2, we introduce the language and models of syntactic epistemic logics with

only the B operator, based on syntactic structures introduced in [12]. We show that formulae in this language are preserved under Σ -isomorphism (truth assignments agreeing on atomic and epistemic formulae in the set Σ) for certain sets Σ . In section 3 we introduce the language and interpretation of the logic with operators $\min(n)$ and show that $\min(n)$ is not definable in the basic language, since $\min(n)$ formulae are not preserved under Σ -isomorphism. We also show how to define $\max(n)$, ∇ and other operators in the extended language. In section 4, we give a sound and (weakly) complete axiomatisation of the logic with $\min(n)$, and prove that its satisfiability problem is NP-complete. We also show that adding $\max(n)$ as an axiom captures exactly the set of models where the agent’s belief set has cardinality of at most n . In section 5 we define a modal logic with $\min(n)$ and ‘next state’ modality \diamond , in which we can express properties relating to dynamics of agent’s beliefs, such as bounded monotonicity: until the agent’s state is ‘full’, it carries all its current beliefs to the next state. We prove soundness and weak completeness for this logic and show that its satisfiability problem is in PSPACE.

2 Syntactic Structures

In this section we introduce the language and models of basic syntactic epistemic logic. Syntactic epistemic logic considers beliefs as syntactic objects rather than propositions (sets of possible worlds). This is done to, e.g., avoid closure under logical consequence and identifying logically equivalent beliefs. We base our account of the interpretations of the language of epistemic logic on *syntactic structures* introduced in [12].

The language \mathcal{L} of basic syntactic epistemic logic is parameterised by a set \mathcal{P} of primitive propositions. \mathcal{L} is defined as follows.

$$\phi ::= p \in \mathcal{P} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B\phi$$

We use the usual derived propositional connectives. The intended meaning of $B\phi$ is that ϕ is believed by the agent. We denote the set of *epistemic atoms* $\{B\phi : \phi \in \mathcal{L}\}$ by \mathcal{B} . The elements of the set $\mathcal{P} \cup \mathcal{B}$ of primitive propositions and epistemic atoms, are called *atoms*.

A *syntactic structure* [12], henceforth sometimes called just ‘a model’, is a pair $M = (S, \sigma)$, where S is a set of *states* and

$$\sigma : S \rightarrow 2^{\mathcal{P} \cup \mathcal{B}}$$

Thus a syntactic structure identifies a set of formulae believed by the agent in a state $s \in S$. We call this set the agent’s *epistemic state* in s , and denote it $\overline{\sigma}(s)$: $\overline{\sigma}(s) = \{\phi : B\phi \in \sigma(s)\}$. A pair M, s , where s is a state of M , is called a *pointed model*. The definition of satisfaction of a formula ϕ in a pointed model M, s , written $M, s \models$

¹ University of Bergen, Norway, agotnes@ii.uib.no

² University of Nottingham, UK, nza@cs.nott.ac.uk

ϕ , is straightforward:

$$\begin{aligned} M, s \models p &\Leftrightarrow p \in \sigma(s) \\ M, s \models B\phi &\Leftrightarrow \phi \in \overline{\sigma}(s) \\ M, s \models \neg\phi &\Leftrightarrow M, s \not\models \phi \\ M, s \models \phi_1 \wedge \phi_2 &\Leftrightarrow M, s \models \phi_1 \text{ and } M, s \models \phi_2 \end{aligned}$$

We remark that in this definition of syntactic knowledge by [12] in a possible worlds framework, the states themselves do not play any important part: the interpretation of a formula in a state s does not depend on any state different from s . Thus, the interpretation in s of any formula can be defined solely in terms of the set $\sigma(s)$. We will, however, make use of the full set of states S when we introduce syntactic relational structures in Section 5.

We use \mathcal{M} to denote the class of all syntactic structures. \mathcal{M} can be seen as models of agents with unbounded memory. In the following, we will also be interested in other classes of syntactic structures. For a given natural number n , \mathcal{M}^n is the class of syntactic structures where at most n formulae are believed at the same time, i.e., where $|\overline{\sigma}(s)| \leq n$ for every $s \in S$. \mathcal{M}^n can be seen as models of agents with a fixed memory size. \mathcal{M}_{fin} denotes the class of syntactic structures with finite epistemic states, i.e., $\mathcal{M}_{fin} = \bigcup_{n \in \mathbb{N}} \mathcal{M}^n$. \mathcal{M}_{fin} can be seen as models of agents with finite memory.

2.1 Preservation

As we will soon extend the logical language, we need to be able to compare the expressiveness of different languages. To do that precisely, we introduce the notion of Σ -isomorphism between two pointed models, where Σ is a set of atoms. Two pointed models are Σ -isomorphic, if they agree on all formulae in Σ . Formally:

Definition 1 Let $M = (S, \sigma)$ and $M' = (S', \sigma')$ be models, $s \in S$ and $s' \in S'$, and $\Sigma \subseteq \mathcal{P} \cup \mathcal{B}$. $(M, s) \sim_\Sigma (M', s')$, (M, s) and (M', s') are Σ -isomorphic, is defined as follows:

$$(M, s) \sim_\Sigma (M', s') \iff \sigma(s) \cap \Sigma = \sigma'(s') \cap \Sigma$$

Given a $\phi \in \mathcal{L}$, the set $Subf(\phi)$ is the set of all subformulae of ϕ (with subformulae of the form $B\psi$ treated as atomic formulae), and $At(\phi)$ is the set of atomic subformulae of ϕ , including epistemic atoms: $At(\phi) = Subf(\phi) \cap (\mathcal{P} \cup \mathcal{B})$.

The following lemma states that the truth value of a formula depends only on the truth value of its atomic subformulae, or, in other words, satisfaction of \mathcal{L} formulae ϕ is invariant under $At(\phi)$ -isomorphism.

Lemma 1 For any two pointed models M, s and M', s' , $\Sigma \subseteq \mathcal{P} \cup \mathcal{B}$ and $\phi \in \mathcal{L}$ such that $At(\phi) \subseteq \Sigma$:

$$(M, s) \sim_\Sigma (M', s') \Rightarrow (M, s \models \phi \Leftrightarrow M', s' \models \phi)$$

The proof is straightforward.

3 Upper/Lower Bounds on Belief Sets

We now extend the language \mathcal{L} with operators for expressing properties of syntactic structures such as ‘‘at most n different formulae are believed’’. Furthermore, we show that an extension of the language indeed was necessary to express such properties.

The language \mathcal{L}_{min} is defined by adding a nullary operator $min(n)$, for each natural number n , to the language \mathcal{L} . Formally \mathcal{L}_{min} is defined as follows:

$$\phi ::= p \in \mathcal{P} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B\alpha : \alpha \in \mathcal{L} \mid min(n) : n \in \mathbb{N}$$

Let $M = (S, \sigma)$ be a model, and $s \in S$. The satisfaction of a \mathcal{L}_{min} formula ϕ in (M, s) is defined by adding the following clause to the definition of satisfaction of \mathcal{L} formulae:

$$M, s \models min(n) \Leftrightarrow |\overline{\sigma}(s)| \geq n$$

A formula $min(n)$ captures the notion that *at least* n formulae are believed. As mentioned above, we are also interested in a dual expressing that *at most* n formulae are believed. The reason that a dual operator to $min(n)$ is not included in \mathcal{L}_{min} , is that it is in fact derivable. We define it as the following derived operator:

$$max(n) \equiv \neg min(n+1)$$

It is easy to see that

$$M, s \models max(n) \Leftrightarrow |\overline{\sigma}(s)| \leq n$$

3.1 Expressive Power

We compare the expressive power of the languages \mathcal{L} and \mathcal{L}_{min} : are, for example, $min(n)$ and/or $max(n)$ expressible in \mathcal{L} ?

It was shown in the previous section that \mathcal{L} formulae are invariant under Σ -isomorphism. On the other hand, as the following lemma shows, in the case of \mathcal{L}_{min} formulae, satisfaction is *not* invariant under Σ -isomorphism³.

Lemma 2 There are pointed models M, s, M', s' and $\Sigma \subseteq \mathcal{P} \cup \mathcal{B}$ such that $(M, s) \sim_\Sigma (M', s')$, but for some $\phi \in \mathcal{L}_{min}$ with $At(\phi) \subseteq \Sigma$

$$M, s \models \phi \text{ and } M', s' \not\models \phi$$

Proof. Take $\phi = min(1)$, $M = (S, \sigma)$, $M' = (S', \sigma')$, $\overline{\sigma}(s) = \{\alpha\}$ for some $\alpha \in \mathcal{L}$, $\overline{\sigma}(s') = \emptyset$ and $\Sigma = \emptyset$. Trivially, $At(\phi) \subseteq \Sigma$. $|\overline{\sigma}(s)| \geq 1$, so $M, s \models \phi$. $|\overline{\sigma}(s')| \not\geq 1$, so $M', s' \not\models \phi$. \square

The following theorem follows immediately from the fact that \mathcal{L}_{min} can discern between Σ -isomorphic models, while \mathcal{L} cannot (Lemmata 1 and 2).

Theorem 1 \mathcal{L}_{min} is strictly more expressive than \mathcal{L} .

3.1.1 Knowing At Least and At Most

In [1, 4] two dual operators were introduced to express properties of syntactic knowledge. Both operators are unary, and take a *finite* set of object formulae as argument (it is thus assumed that the language has symbols for finite sets of formulae). Let $X \subseteq \mathcal{L}$ be finite. First, ΔX is intended to mean that the agent knows *at least* the formulae in the set X : the agent knows every formula in X , but might in addition also know other formulae not in X . Second, ∇X is intended to mean that the agent knows *at most* the formulae in the set X : every formula the agent knows is in X , but it might not know every formula in X . The ∇ operator can thus be seen as a syntactic version, without the assumption that belief is closed under logical consequence, of an ‘‘only knowing’’ operator [15]. ∇X is not definable by B [2]. The

³ Most logics satisfy the principle of locality: the truth value of a formula does not depend on the assignment to variables other than the formula’s free variables. This is such an obvious property that it usually goes unremarked; however some logics do violate it. This phenomenon was investigated for predicate logics in e.g. [16]; for propositional logics, the only example we know of in addition to the logics of ∇ and $min(n)$ is the logic of *only knowing* [15].

conjunction of knowing at least X and knowing at most X , knowing *exactly* X , is written $\boxtimes X$. The \triangleright and \boxtimes operators can be used to express compactly that the agent knows the given formulas *and nothing else*. For example, from the fact that $\boxtimes\{\text{Bird}(\text{Tweety})\}$ we can derive $\neg B \neg \text{Flies}(\text{Tweety})$.

Formally, satisfaction is defined as follows:

$$\begin{aligned} M, s \models \Delta X &\Leftrightarrow X \subseteq \overline{\sigma}(s) \\ M, s \models \triangleright X &\Leftrightarrow \overline{\sigma}(s) \subseteq X \\ M, s \models \boxtimes X &\Leftrightarrow \overline{\sigma}(s) = X \end{aligned}$$

It turns out that both notions of knowing at most and knowing at least a finite set of formulae are definable in \mathcal{L}_{\min} . We leave it to the reader to check that

$$\begin{aligned} \Delta X &\equiv \bigwedge_{\alpha \in X} B\alpha \\ \boxtimes X &\equiv \Delta X \wedge \max(|X|) \\ \triangleright X &\equiv \bigvee_{Y \subseteq X} \boxtimes Y \end{aligned}$$

4 Completeness and Complexity

In this section, we give a complete and sound axiomatisation of the logic of $\min(n)$. Let \mathcal{S} be the logic defined by the following axiom schemata and rules over the language \mathcal{L}_{\min} :

Prop	all substitution instances of propositional logic
MIN0	$\min(0)$
MIN1	$\min(n) \rightarrow \min(m)$ $m < n$
MIN2	$(B\phi_1 \wedge \dots \wedge B\phi_n) \rightarrow \min(n)$ $\forall i \neq j \in [1, n] \phi_i \neq \phi_j$
MP	If $\phi, \phi \rightarrow \psi$ then ψ

It is easy to see that all axioms are valid on all syntactic structures, and that the following holds.

Lemma 3 \mathcal{S} is sound with respect to \mathcal{M} .

Observe that the logic of \mathcal{M} (or of \mathcal{M}_{fin}) is not compact. For example, consider a set of formulae which says that the agent has at least one belief, but it does not believe any formula: $\{\min(1)\} \cup \{\neg B\phi : \phi \in \mathcal{L}\}$. Every finite subset of this set is satisfiable, but the set itself is not. This means that we can at most prove weak completeness.

The remainder of this section consists of constructions and intermediate results leading up to the main completeness results in Theorems 2, 3 and 4. First, some definitions. Given sets of formulae Δ, Ξ , we say that Ξ is Δ -maximal if either $\phi \in \Xi$ or $\neg\phi \in \Xi$ for each $\phi \in \Delta$. Let $Cl(\phi)$ be the closure of $Subf(\phi)$ with respect to single negations and $\min(..)$, namely:

- if $\psi \in Subf(\phi)$, then $\psi \in Cl(\phi)$
- $\min(0) \in Cl(\phi)$
- $\min(|\{B\alpha : B\alpha \in Subf(\phi)\}|) \in Cl(\phi)$
- if $\min(n) \in Cl(\phi)$, then $\min(m) \in Cl(\phi)$, for all m with $0 < m < n$
- if $\psi \in Cl(\phi)$, then $\neg\psi \in Cl(\phi)$ unless $\psi = \neg\chi$ for some χ

Clearly, $Cl(\phi)$ is finite.

To prove completeness, given an \mathcal{S} -consistent formula ϕ we now construct a finite model M_ϕ and show that it satisfies ϕ . When ϕ is an \mathcal{S} -consistent formula, let Γ_ϕ be some $Cl(\phi)$ -maximal \mathcal{S} -consistent subset of $Cl(\phi)$ which contains ϕ (it is easy to prove that such a set exists if ϕ is \mathcal{S} -consistent, just pick one of them). Let $Bel(\Gamma_\phi) = \{\psi : B\psi \in \Gamma_\phi\}$. Let $m_\phi = \max(m : \min(m) \in \Gamma_\phi)$. Since Γ_ϕ is finite and contains $\min(0)$ (by **MIN0** and the fact that $\min(0) \in$

$Cl(\phi)$), such an m_ϕ exists. By **MIN2** and the fact that $\min(|\{B\alpha : B\alpha \in Subf(\phi)\}|) \in Cl(\phi)$, the cardinality of $Bel(\Gamma_\phi)$ is less or equal to m_ϕ .

To build the model M_ϕ , in the case that $|Bel(\Gamma_\phi)| = m_\phi$ we can just let the epistemic state be identical to $Bel(\Gamma_\phi)$; as we show below, it is easy to prove a truth lemma in that case. However, when $|Bel(\Gamma_\phi)| < m_\phi$ (for example, if $\phi = \min(10)$, then $|Bel(\Gamma_\phi)| = 0$ and $m_\phi = 10$), we must pad the epistemic state with $m_\phi - |Bel(\Gamma_\phi)|$ extra formulae. These formulae should not come from $\{\psi : B\psi \in Subf(\phi)\}$, but we have an infinite supply of formulae in \mathcal{L} . So let $k_\phi = m_\phi - |Bel(\Gamma_\phi)|$ and for all $i \in \{1, \dots, k_\phi\}$, choose some (unique) $B\alpha_\phi^i \notin Subf(\phi)$. We are now ready to define M_ϕ . Let $M_\phi = (\{s_\phi\}, \sigma_\phi)$ where σ_ϕ is such that

$$p \in \sigma_\phi(s_\phi) \Leftrightarrow p \in \Gamma_\phi \text{ when } p \in \mathcal{P}$$

$$\sigma_\phi(s_\phi) = \begin{cases} Bel(\Gamma_\phi) & |Bel(\Gamma_\phi)| = m_\phi \\ Bel(\Gamma_\phi) \cup \{\alpha_\phi^1, \dots, \alpha_\phi^{k_\phi}\} & \text{otherwise} \end{cases} \quad (A)$$

Note that in both case (A) and (B), the size of the epistemic state is exactly m_ϕ : $|\sigma_\phi(s_\phi)| = m_\phi$.

Lemma 4 (Truth Lemma) For each $\psi \in Subf(\phi)$,

$$M_\phi, s_\phi \models \psi \Leftrightarrow \psi \in \Gamma_\phi$$

Proof. The proof is by induction over the structure of ψ . The case when p is a propositional variable is immediate. When $\psi = B\alpha$, $\alpha \neq \alpha_\phi^i$ for all $i \in [1, k_\phi]$ since $B\alpha \in Subf(\phi)$, so that case is also immediate. Let $\psi = \min(n)$. $M_\phi, s_\phi \models \psi$ iff $m_\phi \geq n$. For the direction to the right, $\min(m_\phi) \in \Gamma_\phi$, so $\min(n) \in \Gamma_\phi$ for any n such that $m_\phi \geq n$ by **MIN1** (and the fact that $\min(n) \in Cl(\phi)$). For the direction to the left, if $\min(n) \in \Gamma_\phi$, then $n \leq m_\phi$ immediately by definition of m_ϕ . The inductive step (negation and conjunction) is straightforward. \square

Define \max_ϕ as the maximum of $|\{B\alpha : B\alpha \in Subf(\phi)\}|$ and $\max(m : \min(m) \in Subf(\phi))$. The following Lemma, showing that every satisfiable ϕ is satisfied in a model of bounded size – particularly in one where the size of the epistemic state is no greater than \max_ϕ , follows immediately (it is easy to see that $m_\phi \leq \max_\phi$):

Lemma 5 Any \mathcal{S} -consistent formula ϕ is satisfied in a state in a model $M = (\{s\}, \sigma)$ where $|\sigma(s)| \leq \max_\phi$ and $|\sigma(s) \cap \mathcal{P}| \leq |\text{At}(\phi) \cap \mathcal{P}|$.

The following theorem follows immediately from Lemmata 3 and 5.

Theorem 2 \mathcal{S} is sound and weakly complete with respect to \mathcal{M} .

Furthermore, since Lemma 5 shows satisfiability in a model with a *finite* epistemic state, the following also holds.

Theorem 3 \mathcal{S} is sound and weakly complete with respect to \mathcal{M}_{fin} .

We now discuss axiomatisation of the class \mathcal{M}^n . Let $n \in \mathbb{N}$ be fixed. Define \mathcal{S}^n to be \mathcal{S} extended with the axiom $\max(n)$.

Theorem 4 \mathcal{S}^n is sound and weakly complete with respect to \mathcal{M}^n .

Proof. Soundness follows immediately from Lemma 3 and the definition of \mathcal{S}^n . For completeness, let ϕ be a \mathcal{S}^n -consistent formula, and let $\phi' = \phi \wedge \max(n)$. Since ϕ is \mathcal{S}^n -consistent, ϕ' is \mathcal{S} -consistent, so it is satisfied in a state with epistemic state of size no greater than $\max_{\phi'}$. It must be the case that $\neg\min(n+1) \in \Gamma_{\phi'}$

(otherwise $\min(n+1) \in \Gamma_{\phi'}$, since $\min(n+1) \in Cl(\phi')$, and thus $\Gamma_{\phi'}$ would be inconsistent). If $\max_{\phi'} \geq n+1$, $\min(n+1) \in \Gamma_{\phi'}$ by **MIN1** and the fact that $\min(\max_{\phi'}) \in \Gamma_{\phi'}$ which is a contradiction, so $\max_{\phi'} < n+1$. Thus, ϕ' , and therefore also ϕ , are satisfied in a state where the epistemic state is no greater than n . \square

4.1 Complexity

The *satisfiability problem* for \mathcal{S} is the problem of determining, given a formula ϕ , whether there exists a structure M with a state s such that $M, s \models \phi$ (we here abuse the terminology somewhat and use \mathcal{S} to denote not only the axiomatic system but also the logic of M).

To determine the complexity of the satisfiability problem precisely, we need to decide on the encoding of formulas and models. Consider some standard encoding of propositional formulas as strings, for example where propositional variables are encoded by a single symbol p followed by an index of the variable in binary (see, e.g., [7]). We can encode $\min(n)$ in a similar way, as a single symbol m followed by the representation of n in binary. Pointed models, which are essentially just assignments to atoms, can be encoded either as a concatenation of encodings of the atoms which are true under the assignment, or, if it is an assignment to a finite ordered set of m atoms, as a binary string of length m . Note that in either case the length of the encoding of a satisfying model for a formula ϕ guaranteed by Lemma 5 may be exponential in $|\phi|$. For example, if $\phi = \min(n)$, then $|\phi| = 1 + \log_2(n)$ and the length of the encoding of the satisfying assignment is n (under the bit string approach) or $n \log(n)$ (under the list of atoms approach). This is bad news, because from the existence of an exponential satisfying model we can only infer a NEXPTIME upper bound for the complexity of satisfiability. However, we can encode the satisfying model more efficiently, so that the encoding is not exponential in size, but still can be recognised as a model and used to evaluate a formula. Namely, when we are guessing a model M_ϕ for ϕ , with a single state s_ϕ , we will only explicitly represent the assignment to $At(\phi)$ in s_ϕ , which is polynomial in $|\phi|$. Instead of explicitly representing the assignment to ‘padding formulae’, we will guess the total number of epistemic atoms in $\overline{\sigma}(s)$ and write it down in binary. So the representation of s_ϕ will look as follows: $(\alpha_1, \dots, \alpha_m, n)$, where $\alpha_1, \dots, \alpha_m \subseteq At(\phi)$ are the atoms true in s_ϕ , and $n \geq m$ is a binary representation of the size of $\overline{\sigma}(s)$. Clearly, this is polynomial in $|\phi|$, and this information is sufficient to evaluate ϕ , which can be done in polynomial time.

Theorem 5 *The satisfaction problem for \mathcal{S} is NP-complete.*

Proof. We know that ϕ is satisfiable iff it is satisfiable in a model of size at most \max_{ϕ} , by Lemma 5. We guess this model M_ϕ , and represent it as a string linear in the size of $|\phi|$, by explicitly representing truth values for atoms in $At(\phi)$, and guessing the total number of true epistemic atoms, written in binary. We can check whether ϕ is satisfied by the model in polynomial time. NP-hardness follows from the fact that the logic extends propositional logic. \square

5 Adding State Transitions

In this section we consider the dynamics of the agent’s beliefs. We extend syntactic structures to include state transitions, and the languages \mathcal{L} and \mathcal{L}_{\min} to include a ‘next state’ modality \diamond , to obtain the languages \mathcal{L}^\diamond and $\mathcal{L}_{\min}^\diamond$, respectively; a formal definition is given

below. Extending the logic this way allows us to describe how the agent’s beliefs change. For example, if the agent knows an inference rule, then it can apply it to beliefs in its current state to derive new beliefs which are added to the next epistemic state. To use an example from [12], suppose the agent knows $a = b$ and $b = c$, and is capable of reasoning about equality. Unlike in [12], we do not assume that in this case the agent can derive $a = c$ instantaneously. Rather, we interpret ‘being able to reason about equality’ as being able to derive new statements about equality which follow from the current beliefs, in some future state. In particular, the agent should be able to reach a state where it believes $a = c$. This property is expressible by the following \mathcal{L}^\diamond formula, in which \diamond stands for ‘there exists a successor state where...’:

$$B(a = b) \wedge B(b = c) \rightarrow \diamond B(a = c)$$

We may also want to express that the agent’s knowledge grows monotonically, which can be done by adding an axiom schema

$$B\phi \rightarrow \square B\phi$$

(where \square is the dual of \diamond , meaning ‘in every successor state...’).

The expressive power we get in \mathcal{L}^\diamond is similar to, for example, step logics [10]: we can describe how the beliefs of an agent capable of applying certain inference rules increase over time. Imposing some simple conditions would guarantee that the set of beliefs remains finite. One of such possible conditions is requiring that each state transition corresponds to deriving exactly one new formula; this is expressible by a schema:

$$\diamond(B\phi \wedge B\psi) \rightarrow (B\phi \vee B\psi)$$

(if in the next state the agent believes two formulae, then at least one of those formulae is already believed in the current state). Note that the condition that each successor state has at most one extra formula can be more elegantly expressed in $\mathcal{L}_{\min}^\diamond$:

$$\max(n) \rightarrow \square \max(n+1)$$

Another useful property, namely that each state transition adds some new belief, is not expressible in \mathcal{L}^\diamond at all, unless we introduce existential quantification over formulae or infinite disjunctions; however in $\mathcal{L}_{\min}^\diamond$ we can say

$$\min(n) \rightarrow \square \min(n+1)$$

If we do take the size bound on the agent’s epistemic state seriously, however, the combination of monotonicity and ability to derive new formulae becomes problematic. A natural restriction on monotonicity in this case would be to say: if the set of beliefs is less than the maximal size, then monotonicity holds; otherwise, the agent can still derive a new formula, but at the expense of ‘overwriting’ one of the old beliefs. This assumption is made, e.g., in [5], which studies logics for bounded memory reasoners. The property of bounded monotonicity (if the cardinality of the set of beliefs is less than n , then all beliefs persist into the next state) can be expressed in $\mathcal{L}_{\min}^\diamond$ as

$$\max(n-1) \wedge B\phi \rightarrow \square B\phi$$

Hopefully, the examples above have given the reader a flavour of the kind of properties we would like to express in $\mathcal{L}_{\min}^\diamond$. Now we proceed to give formal definitions of the language and the structures.

Let the language $\mathcal{L}_{\min}^\diamond$ is defined by the following grammar:

$$\phi ::= p \in \mathcal{P} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid B\phi : \phi \in \mathcal{L} \mid \min(n) : n \in \mathbb{N} \mid \diamond\phi$$

Define $\Box\phi$ as $\neg\Diamond\neg\phi$. A *syntactic relational structure* is a triple $M = (S, \sigma, R)$ where (S, σ) is a syntactic structure and $R \subseteq S \times S$ a relation over the states. The class of all syntactic relational structures is denoted \mathcal{M}^\diamond . The satisfaction relation $M, s \models \phi$ is defined as before; the extra clause for $\Diamond\phi$ is standard in modal logic:

$$M, s \models \Diamond\phi \Leftrightarrow \exists s' (R(s, s') \text{ and } M, s' \models \phi)$$

Let \mathcal{K}_{min} be the logical system obtained by adding to \mathcal{S} the axiom schema \mathbf{K} : $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$ and the necessitation rule \mathbf{N} : $\vdash \phi \Rightarrow \vdash \Box\phi$.

Theorem 6 \mathcal{K}_{min} is sound and weakly complete wrt. \mathcal{M}^\diamond .

Proof. The proof consists of minor modifications of the constructions and proof in Section 4. The construction of a satisfying model $M \in \mathcal{M}^\diamond$ for a \mathcal{K}_{min} -consistent formula ϕ uses a method described in [7].

Define $Cl(\phi)$ as in Section 4. Construct $M = (S, \sigma, R)$ as follows. Let S be the set of ($Cl(\phi)$ -) maximal consistent subsets of $Cl(\phi)$. In Section 4 it was enough to define a satisfying assignment in a single state s_ϕ corresponding to the maximal consistent set Γ_ϕ containing ϕ ; here we must repeat that exercise for *any* such set. For each $s \in S$, the assignment $\sigma(s)$ is defined from s in exactly the same way as $\sigma_\phi(s_\phi)$ was defined from Γ_ϕ in Section 4. Note that we do not include the ‘padding’ formulae $B\alpha_1, \dots, B\alpha_k$ in the state s ; they are just a technical device we use to define the assignment. In particular, the statement of the Truth Lemma is restricted to the subformulas of ϕ .

Finally, let $R(s, t)$ hold if, and only if, $\phi_s \wedge \Diamond\phi_t$ is \mathcal{K}_{min} -consistent, where ϕ_s (ϕ_t) is the conjunction of formulae in $\sigma(s)$ ($\sigma(t)$). The proof of the Truth lemma is standard, see e.g. [7]. \square

Since the model we have constructed is exponential in length of the encoding of a formula ϕ , the theorem above implies that \mathcal{K}_{min} is decidable in NEXPTIME. However, it is possible to give a tighter upper bound, namely PSPACE:

Theorem 7 The problem of whether a formula $\phi \in \mathcal{L}_{min}^\diamond$ is satisfied in a model in \mathcal{M}^\diamond is PSPACE complete.

Proof. It is easy to show that every satisfiable formula ϕ has a tree model of polynomial depth (the proof is the same as for basic modal logic K, see for example [7]). Each state in the model has polynomial size, as we have shown in Lemma 5. Hence, each branch of the satisfying model can be encoded as a string whose length is polynomial in the length of the encoding of ϕ . The *Witness* algorithm given in [7] which essentially builds a tableaux for ϕ one branch at a time, can be easily adapted to check for satisfiability in \mathcal{M}^\diamond . For PSPACE-hardness, observe that the satisfiability problem for modal logic K can be reduced to the satisfiability problem for \mathcal{K}_{min} , and the former is PSPACE-complete [7]. \square

6 Conclusions

We have modeled the beliefs of an agent with the help of syntactic assignments, an approach also used by several others [14, 11, 6, 4] in order to model properties of reasoners which are difficult or impossible to model with traditional modal epistemic logics. The same model of belief we have used in this paper, was recently used [3] to give a semantics to Ho Ngoc Duc’s [8] logic of rational, but not logically omniscient, agents. While many approaches have been suggested to alleviate the logical omniscience problem [13] the syntactic approach can be seen as the most general one. The results in this

paper should be readily applicable to other logics with a syntactic component. Of particular interest for future work would be to apply them to the logic of general awareness [11].

In the language of syntactic epistemic logic with $\min(n)$ operators, we can express many interesting properties of agents, such as bounded memory, and knowing exactly or at most the given set of formulae and nothing else. We introduce several natural logics in this language, and show that they have sound and complete axiomatisations, and that their decidability problem is in the same class as their non-epistemic counterparts.

Acknowledgements Thomas Ågotnes’ work was supported by NFR grant 166525/V30. The initial phases of this work were carried out while Thomas Ågotnes was visiting the intelligent agents lab in the School of Computer Science at the University of Nottingham. Natasha Alechina would like to thank the Royal Society for sponsoring related work on logics for resource-bounded agents, Isaac Newton Institute for hospitality during work on this paper, and Andrei Krokhin for discussing complexity.

REFERENCES

- [1] T. Ågotnes, *A logic of Finite Syntactic Epistemic States*, Ph.D. thesis, Department of Informatics, University of Bergen, Norway, April 2004.
- [2] T. Ågotnes and N. Alechina, ‘The dynamics of syntactic knowledge’, Technical Report 304, Dept. of Informatics, Univ. of Bergen, Norway, (2005).
- [3] T. Ågotnes and N. Alechina, ‘Semantics for dynamic syntactic epistemic logics’, in *Principles of Knowledge Representation and Reasoning: Proceedings of the Tenth International Conference (KR’06)*. To appear.
- [4] T. Ågotnes and M. Walicki, ‘Strongly complete axiomatizations of “knowing at most” in standard syntactic assignments’, in *Sixth International Workshop, CLIMA VI, City University London, UK, June 27-29, 2005. Revised Selected and Invited Papers.*, eds., F. Toni and P. Torroni, volume 3900 of *Lecture Notes in Artificial Intelligence (LNAI)*, pp. 57–76. Heidelberg: Springer-Verlag, (2006).
- [5] A. Albores, N. Alechina, P. Bertoli, C. Ghidini, B. Logan, and L. Serafini, ‘Model-checking memory requirements of resource-bounded reasoners’, in *Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI)*. AAAI Press, (2006). To appear.
- [6] N. Alechina, B. Logan, and M. Whitsey, ‘A complete and decidable logic for resource-bounded agents’, in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pp. 606–613. ACM Press (2004).
- [7] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, Cambridge University Press, 2001.
- [8] Ho Ngoc Duc, ‘Reasoning about rational, but not logically omniscient, agents’, *Journal of Logic and Computation*, 7(5), 633–648, (1997).
- [9] J. Elgot-Drapkin, M. Miller, and D. Perlini, ‘Memory, reason and time: the Step-Logic approach’, in *Philosophy and AI: Essays at the Interface*, 79–103, MIT Press, Cambridge, Mass., (1991).
- [10] J. Elgot-Drapkin and D. Perlini, ‘Reasoning situated in time I: Basic concepts’, *Journal of Experimental and Theoretical Artificial Intelligence*, 2, 75–98, (1990).
- [11] R. Fagin and J. Y. Halpern, ‘Belief, awareness and limited reasoning’, *Artificial Intelligence*, 34, 39–76, (1988).
- [12] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning About Knowledge*, The MIT Press, Cambridge, Massachusetts, 1995.
- [13] J. Hintikka, ‘Impossible possible worlds vindicated’, *Journal of Philosophical Logic*, 4, 475–484, (1975).
- [14] K. Konolige, *A Deduction Model of Belief*, Morgan Kaufmann Publishers, Los Altos, California, 1986.
- [15] H. J. Levesque, ‘All I know: a study in autoepistemic logic’, *Artificial Intelligence*, 42, 263–309, (1990).
- [16] I. Németi, ‘Fine-structure analysis of first-order logic’, in *Arrow logic and multi-modal logic*, eds., M. Masuch and L. Pólos, 221 – 247, CSLI Publications, Stanford, (1996).

Modal logics for communicating rule-based agents

Natasha Alechina and Mark Jago and Brian Logan¹

Abstract. In this paper, we show how to establish correctness and time bounds (e.g., quality of service guarantees) for multi-agent systems composed of communicating rule-based agents. The formal models of multi-agent systems we study are transition systems where each transition corresponds to either a rule firing or an act of communication by an agent. We present a complete and sound modal logic which formalises how the beliefs of communicating rule-based agents change over time. Using a simple example, we show how this logic can be used to specify temporal properties of belief change in multi-agent systems in a precise and realistic way, and how existing modal logic techniques such as model-checking can be used to state and verify properties of agents.

1 INTRODUCTION

There has recently been considerable interest in rule-based approaches to various aspects of agent technology. For example, rules and rule engines (rule interpreters) have been advocated as a means of managing the business logic of applications in a wide range of domains such as insurance, financial services, government, telecom customer care and billing, and ecommerce [19]. Another important application of rule-based approaches in agent-based systems is the semantic web, for example, ontological reasoning in DAML/OWL and, more generally, in rule extensions to ontologies such as OWL+Rules [17] and SWRL[18], which significantly increase the expressive power of ontology languages [17].

At the same time there has been a growing use of rule engines as key components of agent based systems. Much of this work is based on mature rule-based systems technology such as the Jess rule engine [14], e.g., the FIPA-OS JessAgent, the use of Jess in reasoning about ontologies (DAML JessKB) and in web service composition (DAML-S Virtual Machine [20]) and supporting tools such as Jena which convert XML/RDF to Jess facts, and, more generally, the Java Rules API.² However, while the adoption of rule-based approaches brings great benefits in terms of flexibility and ease of development, these approaches also raise new challenges for the agent developer, namely how to ensure correctness of rule-based designs (will a rule-based agent produce the correct output for all legal inputs), termination (will a rule-based agent produce an output at all) and response time (how much computation will a rule-based agent have to do before it generates an output).

As a motivating example, consider an agent which acts as a provider agent for a company which sells goods or services on the web. Such an agent may require a combination of ontological rules to match a potential customer's request against the goods and services it offers (e.g., to realise that dog food is a kind of pet food

[21]) and business rules to determine, e.g., the appropriate level of discount to be offered. The designer or developer of such an agent may wish to ensure that the agent's behaviour is correct, e.g., that the agent does not offer a customer an inappropriate discount, or accept an order from a customer with an unpaid bill, and/or may wish to offer certain quality of service guarantees, e.g., to bound the time a potential customer has to wait between the acknowledgement of a request for a price and receiving a quote.

The upper limit on deliberation (or response) time in rule-based systems is a well-established problem. However previous work has studied expert and diagnostic systems as single isolated systems [15], and has focused mainly on termination (or worst case response time), rather than more general issues of correctness and quality of service. In this paper, we show how to establish correctness and time bounds for multi-agent systems composed of rule-based agents. The formal models of multi-agent systems we study are transition systems where each transition corresponds to a rule firing by an agent (or an act of communication, which we also model as an effect of a rule firing). To solve a particular problem or to achieve a particular state, agents typically need to fire multiple rules; the number of transitions until the goal state is reached gives an estimate of the time required for deliberation. We present a modal epistemic logic where the agent's rules and its knowledge in each state determine conditions on the accessibility relation in the model. The novelty of our work is in representing a multi-agent system of communicating rule-based agents as a transition system, and proposing a modal logic to describe these systems. In addition to being able to state time-related properties of the system, this also enables us to use existing modal logic techniques such as model-checking to state and verify properties of agents.

The remainder of the paper is organised as follows: in section 2 we introduce our model of communicating rule-based agents. In sections 3, 4 and 5 we introduce the language and its semantics, present a complete and sound axiomatisation of our logic, and state decidability. In section 7 we briefly illustrate the logic by giving an example and show how properties expressed in the logic can be verified using an existing model checker, before discussing related work in section 8.

2 COMMUNICATING RULE-BASED AGENTS

We assume that each agent has a *program*, consisting of Horn clause rules, and a working memory, which contains facts. The set of rules comprising the agent's program and the facts contained in the agent's working memory together constitute the agent's beliefs.

To define the agent's internal language more precisely, we fix a set of predicate symbols \mathcal{P} , a set of variables \mathcal{X} and a set of constants \mathcal{D} . A literal λ is a predicate symbol of n arguments followed by n variables or constants and possibly preceded by a negation symbol ' \neg '. For example, *LuxuryProduct(x)* and

¹ School of Computer Science, University of Nottingham, Nottingham, NG1 1BB, UK, email: {nza,mtw,bsl}@cs.nott.ac.uk

² Jess is the reference implementation for the Java Rules API.

$\text{LuxuryProduct(Porsche)}$ are both literals. When every argument of the predicate symbol in a literal is an element of \mathcal{D} , e.g., $\text{LuxuryProduct(Porsche)}$, that literal is called a *ground literal*. Each rule-based agent has a finite set \mathcal{R} of rules, which are of the form

$$\lambda_1, \dots, \lambda_n \rightarrow \lambda$$

where λ_i, λ are literals. λ is called the *consequent* of the rule (denoted by $\text{cons}(\lambda_1, \dots, \lambda_n \rightarrow \lambda)$), and each λ_i is an *antecedent* of the rule. An example of a rule is:³

$$\begin{aligned} \text{PremiumCustomer}(x), \text{LuxuryProduct}(y) \rightarrow \\ \text{Discount}(x, y, 7.5\%) \end{aligned}$$

We assume that the rules do not contain functional symbols. Given a rule $\lambda_1, \dots, \lambda_n \rightarrow \lambda$, we denote an *instance* of this rule as $\delta(\lambda_1, \dots, \lambda_n \rightarrow \lambda)$, where δ is some substitution function from the set of variables of the rule into \mathcal{D} . For example, if δ assigns *Miller* to x and *Porsche* to y , then

$$\begin{aligned} \delta(\text{PremiumCustomer}(x), \text{LuxuryProduct}(y) \rightarrow \\ \text{Discount}(x, y, 7.5\%)) \\ = \\ \text{PremiumCustomer(Miller)}, \text{LuxuryProduct(Porsche)} \rightarrow \\ \text{Discount}(Miller, Porsche, 7.5\%) \end{aligned}$$

A rule $\lambda_1, \dots, \lambda_n \rightarrow \lambda$ *matches* if there is a substitution δ such that the agent's working memory contains $\delta(\lambda_1), \dots, \delta(\lambda_n)$. *Firing* the matching rule instance $\delta(\lambda_1, \dots, \lambda_n \rightarrow \lambda)$ adds the ground literal $\delta(\lambda)$ to the agent's working memory. In what follows, we assume that the agents fire at most one rule per cycle (we discuss other rule application strategies in section 6), and that rule matching is refractory, i.e., that each rule instance is fired at most once.

Consider a set $\mathcal{A} = \{1, \dots, n\}$ of n agents. To model communication between agents, we assume that agents have special communication constructs in their language: ‘tell(i, j) λ ’, which we will refer to as a *tell* and ‘ask(i, j) λ ’, which we will refer to as an *ask*. In both cases, i and j are agents and λ is a literal not containing an ask or a tell. tell(i, j) λ stands for ‘ i tells j that λ ’ and ask(i, j) λ stands for ‘ i asks j whether λ is the case’. The position in which these formulas may appear in a rule depends on which agent's program the rule belongs to. Agent i may have an ask or a tell with arguments (i, j) , in the *consequent* of a rule, e.g.:

$$\lambda_1, \dots, \lambda_n \rightarrow \text{ask}(i, j)\lambda$$

Whereas agent j may have the same expressions in the *antecedent* of the rule. For example:

$$\text{tell}(i, j)\lambda \rightarrow \lambda$$

is a well-formed rule for agent j which makes it trust i when i informs it that λ is the case. No other occurrences of tell(i, j) and ask(i, j) are allowed. In particular, i must be distinct from j in all the cases just listed, for our agents neither ask nor tell themselves anything.

When a rule has either an ask or a tell as its consequent, we call it a *communication rule*, or *c-rule* for short. All other rules are known as *deduction rules*, or *d-rules*. These include rules with asks and tells in the antecedent (as well as rules containing neither an ask nor a tell).

³ The business rules in the running example are taken from the RuleML tutorial [8].

Firing a *c-rule* instance with the consequent tell(i, j) λ adds the literal $\delta(\text{tell}(i, j)\lambda)$ both to the working memory of i and of j . Intuitively, i has a record that it told j that $\delta(\lambda)$, and j has a record of being told by i that $\delta(\lambda)$. Similarly, if the consequent of a *c-rule* instance is of the form $\delta(\text{ask}(i, j)\lambda)$, then the corresponding ask is added to the working memories of both i and j .

In this paper, for simplicity we do not consider the interaction between agents and their environment. However we could model the environment as a distinguished agent, e.g., with a different rule application strategy, and interpret tells from the agents to the environment as actions, and tells from the environment to the agents as sensing.

3 LANGUAGE

First we define agent i 's internal language \mathcal{L}_i . For simplicity, we assume that all agents have the same finite set of predicate symbols \mathcal{P} and a finite set of constants \mathcal{D} (although this assumption is not essential). We denote the set of all possible substitutions $\delta : \mathcal{X} \longrightarrow \mathcal{D}$ by Σ . Note that given finite sets \mathcal{X} and \mathcal{D} , the set of all possible substitutions Σ and the set of all possible rule instances are finite as well. Literals are denoted by $\lambda, \lambda_1, \lambda_2, \dots, \text{tell}(i, j)\lambda, \text{ask}(i, j)\lambda, \dots$ ($i, j \in \mathcal{A}$), ground literals are denoted by $\delta(\lambda), \delta(\lambda_1), \dots, \text{tell}(i, j)\delta(\lambda), \text{ask}(i, j)\delta(\lambda), \dots$, where $\delta \in \Sigma$, and rules of the form $\lambda_1, \dots, \lambda_n \rightarrow \lambda$ are denoted by $\rho, \rho_1, \rho_2, \dots$. Note that only rules and ground literals are formulas of \mathcal{L}_i , and that rules are well-formed only if the conditions on the occurrences of asks and tells are satisfied.

In the modal language \mathcal{ML} over \mathcal{L}_i , we have a belief operator B_i for each agent $i \in \mathcal{A}$. The primitive wffs of $\mathcal{ML}(\mathcal{P}, \mathcal{D})$ are:

$$B_i \delta(\lambda) \mid B_k \text{tell}(i, j)\delta(\lambda) \mid B_k \text{ask}(i, j)\delta(\lambda) \mid B_i \rho$$

where $\delta(\lambda)$ is a ground literal, i and j are two different agents (we assume that $|\mathcal{A}| \geq 2$), $k \in \{i, j\}$, and ρ is a well-formed rule of agent i . If ϕ_1 and ϕ_2 are both $\mathcal{ML}(\mathcal{P}, \mathcal{D})$ wffs, the complex wffs of $\mathcal{ML}(\mathcal{P}, \mathcal{D})$ are then given by

$$\neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \diamond\phi_1$$

where \diamond is the next state modality; \vee and \rightarrow are defined as usual, and $\square\phi =_{df} \neg\diamond\neg\phi$.

4 SEMANTICS

In this section we define the transition systems we use to interpret the language introduced above. The states of the transition system are tuples of local states of the agents and are used to interpret the basic belief formulas of the language; intuitively, $B_i \alpha$ is true if the formula α is either a rule of agent i 's program, or is contained in the agent's local state (working memory) in that state. Transitions between states correspond to the agents firing a single rule instance each, in parallel. In the case in which an agent fires an instance of a *d-rule*, a single new formula is added to the agent's state. In the case in which an agent fires an instance of a *c-rule* involving another agent j , the conclusion of the rule is added both to the agent's state and to the agent j 's state.

The formal definition of models is as follows.

Definition 1 (Models) Given a group of agents $\mathcal{A} = \{1, \dots, n\}$, a multi-agent model M is an $n + 3$ -tuple

$$\langle S, \mathcal{A}, T, V_1, \dots, V_n \rangle$$

where S is a set of states, T is the transition relation and each $V_i : S \rightarrow \wp(\mathcal{L}_i)$ is the labelling function for agent $i \in \mathcal{A}$ assigning a set of \mathcal{L} -formulas to each state.

We say that a rule $\lambda_1, \dots, \lambda_n \rightarrow \lambda$ is s - δ - i -applicable if s is a state, δ a substitution, i an agent, $\delta(\lambda_1), \dots, \delta(\lambda_n) \in V_i(s)$ and $\delta(\lambda) \notin V_i(s)$.

The following conditions on the assignments V_i (for all $i \in \mathcal{A}$) and the accessibility relation T hold in all models:

1. for every $i \in \mathcal{A}$, any two states $s, s' \in S$, and every rule formula $\rho, \rho \in V_i(s)$ if, and only if, $\rho \in V_i(s')$
2. For any two states s and s' , $T(s, s')$ holds if, and only if, for every agent i , $V_i(s') = V_i(s) \cup \{\delta_i(\text{cons}(\rho_i))\} \cup C_i$, where ρ_i is some s - δ_i - i -applicable rule if such a rule exists (otherwise $\{\delta_i(\text{cons}(\rho_i))\} = \emptyset$), and $C_i = \{\delta_j \text{ cons}(\rho_j) : \delta_j \text{ cons}(\rho_j)$ is of the form $\text{tell}(j, i)\alpha$ or $\text{ask}(j, i)\alpha\}$.

Condition 1 says that the agent's program does not change, and condition 2 says that each agent fires a single applicable rule instance if it has one, otherwise its state does not change apart from possibly as a result of communication from other agents.

A state transition system can be visualised as follows. Each state contains n finite sets of formulas, corresponding to the beliefs of each of the n agents. Each applicable rule instance of agent i corresponds to a possible ‘action’ by i , namely the addition of the consequent of the rule to the next state. A transition by the system is an n -tuple of such actions (if agent j has no applicable rule instance, the j th component of the tuple is a null action). If in a state s , each agent i has k_i different applicable rule instances, then there are $k_1 \times \dots \times k_n$ different transitions from s , one for every possible combination of agent ‘actions’. This also means that on different execution paths originating in s , applicable rule instances of agent i may be fired in a different order. Suppose there are two such rule instances in s , $\delta_1(\rho_1)$ and $\delta_2(\rho_2)$. Then there is a path to some state s' reached by agent i firing $\delta_1(\rho_1)$ (and other agents firing their rule instances) and from s' to s'' where i gets to fire $\delta_2(\rho_2)$. There is also a path where the order in which $\delta_1(\rho_1)$ and $\delta_2(\rho_2)$ are fired is reversed, and it is also possible that in the next state agent i will acquire other applicable rule instances which on some execution paths will be fired earlier than the one instance remaining from s . If none of the agents has an applicable rule instance, then only one transition is possible (a null action by all agents), to the same state.

The relation of a formula ϕ being satisfied in a model M ($M \models \phi$) is defined as follows:

$$\begin{aligned} M, s \models B_i \alpha &\text{ iff } \alpha \in V_i(s), \text{ for } i \in \mathcal{A}, \alpha \in \mathcal{L}(\mathcal{P}, \mathcal{D}) \\ M, s \models \neg\phi &\text{ iff } M, s \not\models \phi \\ M, s \models \phi \wedge \psi &\text{ iff } M, s \models \phi \text{ and } M, s \models \psi \\ M, s \models \diamond\phi &\text{ iff there is a } u \in S \text{ such that } T(s, u) \text{ and } M, u \models \phi \end{aligned}$$

Although our agents share a common language, each has its own program. Given a program (set of rules) \mathcal{R}_i for each agent $i \in \mathcal{A}$, we define the *program set* $\mathbb{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_n\}$. We want to pick out those models in which agents believe all the rules in their program and no other rules. Such models comprise the class $\mathbf{M}_{\mathbb{R}}$.

Definition 2 (The class $\mathbf{M}_{\mathbb{R}}$) A model $M \in \mathbf{M}_{\mathbb{R}}$, where $\mathbb{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_n\}$, iff for every state s in M , $M, s \models B_i \rho$ iff $\rho \in \mathcal{R}_i$. Given a set of \mathcal{ML} -formulas Γ and an \mathcal{ML} -sentence ϕ , we write $\Gamma \Vdash_{\mathbb{R}} \phi$ iff every model of Γ which is in the class $\mathbf{M}_{\mathbb{R}}$ is also a model of ϕ .

We now ask, given a set of programs $\mathbb{R} = \{R_1, \dots, R_n\}$ for n agents in the language $\mathcal{L}(\mathcal{P}, \mathcal{D})$, what logic corresponds to the class $\mathbf{M}_{\mathbb{R}}$?

5 AXIOMATISATION

The logic $\Lambda_{\mathbb{R}}$ over \mathcal{ML} is defined as follows. First, fix a set of agents $\mathcal{A} = \{1, \dots, n\}$, then fix a program set $\mathbb{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_n\}$. Then we define

$$\delta \text{ app}_i(\lambda_1, \dots, \lambda_n \rightarrow \lambda) \stackrel{\text{df}}{=} B_i \delta(\lambda_1) \wedge \dots \wedge B_i \delta(\lambda_n) \wedge \neg B_i \delta(\lambda)$$

and $\text{app}(\rho) \stackrel{\text{df}}{=} \bigvee_{\delta \in \Sigma} \delta \text{ app}_i(\rho)$, where $\rho \in R_i$. The logic $\Lambda_{\mathbb{R}}$ has the following axiom schemata and rules:

C1 all classical propositional tautologies

$$\mathbf{K} \quad \square(\phi \rightarrow \psi) \rightarrow (\square\phi \rightarrow \square\psi)$$

A1 $B_i \rho$, where $\rho \in \mathcal{R}_i$ (agent i believes its rules)

A2 $\neg B_i \rho$, where $\rho \notin \mathcal{R}_i$ (agent i only believes its rules)

A3 $B_i \alpha \rightarrow \square B_i \alpha$ (agents are monotonic reasoners)

A4 $B_i(\lambda_1, \dots, \lambda_n \rightarrow \lambda) \wedge B_i \delta(\lambda_1) \wedge \dots \wedge B_i \delta(\lambda_n) \rightarrow \diamond B_i \delta(\lambda)$

for each $\delta \in \Sigma$ (if a rule matches, its consequent belongs to some successor state)

A5 $\diamond(B_i \alpha \wedge B_i \beta) \rightarrow (B_i \alpha \vee B_i \beta)$

where α and β are not of the form $\text{tell}(j, i)\lambda$ or $\text{ask}(j, i)\lambda$ (at most one new belief of agent i is added in each transition as a result of agent i firing a rule)

A6 $\diamond B_i \alpha \rightarrow (B_i \alpha \vee \bigvee_{\lambda_1, \dots, \lambda_n \rightarrow \lambda \in \mathcal{R}_i, \delta(\lambda)=\alpha} B_i \delta(\lambda_1) \wedge \dots \wedge B_i \delta(\lambda_n))$

where α is not of the form $\text{tell}(j, i)\beta$ or $\text{ask}(j, i)\beta$ (new beliefs of agent i only arise as a result of firing a rule of i , unless it is a communication from another agent j)

A7 $\delta_1^1 \text{ app}_1(\rho_1^1) \wedge \dots \wedge \delta_{k_1}^1 \text{ app}_1(\rho_{k_1}^1) \wedge \dots \wedge \delta_1^n \text{ app}_n(\rho_1^n) \wedge \dots \wedge \delta_{k_n}^n \text{ app}_n(\rho_{k_n}^n) \wedge \dots \wedge \delta_{\delta(\rho)}^{\delta(\rho)} \text{ app}_{\delta(\rho)}(\rho) \rightarrow \neg \delta \text{ app}_1(\rho) \wedge \dots \wedge \neg \delta \text{ app}_n(\rho) \wedge \dots \wedge \neg \delta \text{ app}_{\delta(\rho)}(\rho) \rightarrow \square \bigvee_{f(i) \in \{1, \dots, k_i\}} (B_1 \delta_{f(1)}(\text{cons}(\rho_{f(1)})) \wedge \dots \wedge B_n \delta_{f(n)}(\text{cons}(\rho_{f(n)})))$

for each possible set of applicable rule instances $\{\delta_1^i(\rho_1^i), \dots, \delta_{k_i}^i(\rho_{k_i}^i)\}$ for each agent i , provided this set is non-empty for at least one of the agents. This axiom constrains all possible successors of each state to states resulting from firing applicable rule instances in parallel by all agents.

A8 $\bigwedge_{\rho \in \mathcal{R}} \neg \text{app} \rho \rightarrow \diamond \bigwedge_{\rho \in \mathcal{R}} \neg \text{app} \rho$ (the terminating state has a transition to itself)

A9 $B_i \text{ tell}(i, j)\lambda \leftrightarrow B_j \text{ tell}(i, j)\lambda$ (communication involving tells is perfect)

A10 $B_i \text{ ask}(i, j)\lambda \leftrightarrow B_j \text{ ask}(i, j)\lambda$ (communication involving asks is perfect)

MP From ϕ and $\phi \rightarrow \psi$ derive ψ

N From ϕ derive $\square\phi$

The notion of derivation is standard.

Theorem 1 $\Lambda_{\mathbb{R}}$ is sound and complete for $\mathbf{M}_{\mathbb{R}}$: $\Gamma \vdash_{\mathbb{R}} \phi$ iff $\Gamma \Vdash_{\mathbb{R}} \phi$.

Due to the lack of space, we can only sketch a proof here. Soundness is proved as usual by induction of the length of a derivation. For completeness, we build a canonical model for a $\Lambda_{\mathbb{R}}$ -consistent set of formulas and prove that this model is in $\mathbf{M}_{\mathbb{R}}$.

Theorem 2 The satisfiability problem for $\mathbf{M}_{\mathbb{R}}$ is decidable.

The proof is omitted due to lack of space; it uses filtration to prove that every satisfiable formula has a model of size bounded by the size of the formula and the size of \mathbb{R} .

6 ADDITIONAL AXIOMS

In this section, we discuss additional axiom schemata which we can use to express interesting properties of agents.

First, we consider properties relating to agent communication. The property of agent 1 trusting agent 2 with respect to some literal λ can be expressed as a *c-rule* belonging to agent 1. This is reflected in the following axiom:

Trust(1,2, λ) $B_1(\text{tell}(2, 1)\lambda \rightarrow \lambda)$

The following axiom says that agent 1 is cooperative in that it believes that it should answer agent 2's queries concerning λ , if it already believes that λ :

Answer(1,2, λ) $B_1(\text{ask}(2, 1)\lambda, \lambda \rightarrow \text{tell}(1, 2)\lambda)$

We can state more general properties of trust and cooperativeness by replacing the axioms above by axiom schemas: e.g., trust any agent i rather than a particular agent, and do so with respect to all formulas, rather than a particular predicate.

Another kind of property we can express relates to the rule application strategy of the agents. The logic described above is based on the assumption that agents may fire matching rule instances in any order. This is not a very realistic assumption (although it still allows us to establish an upper bound on when a certain belief will be derived, because all possible rule orderings will include the actual one). For example, an agent's rules (and rule instances) may be ordered, or more recent information may be acted upon first, or queries answered in preference to applying internal deduction rules. Logics corresponding to an ordered rule application strategy were studied in detail in [5]; unfortunately, rule ordering in general cannot be captured by a single, simple axiom schema.

However, there are some rule application strategies which are relatively simple to capture in modal logic. For example, the ‘all rules at each cycle’ strategy can be easily modelled.⁴ This strategy requires the agent to apply all of its rules to all of its ground beliefs in order to transit to the next state. Note that rules are not applied immediately to the *consequences* of matching rule instances; the rules are applied only once in each transition. To model the ‘all rules at each cycle’ strategy, we remove axiom **A5** (which says that at most one new formula is derived) and add the axiom schema which says that there is at most one successor to each state: $\Diamond\phi \rightarrow \Box\phi$. Note that **A7** then becomes redundant.

7 EXAMPLE

In this section we show how the logic defined above can be used to specify temporal properties of belief change in multi-agent systems in a precise and realistic way. We then go on to show how these properties can be verified using an existing model checker.

Consider a simple multi-agent system which implements the business rules example introduced in section 2. Suppose there are two agents involved in giving a quote. Agent 1 is in charge of quoting for products, and has data relating to current prices and the following discount rules:

⁴ This is also the strategy that is used in step logic [13].

R0 $\text{Quote}(x, y) \rightarrow \text{ask}(1, 2)\text{PremiumCustomer}(x)$

R1 $\text{tell}(2, 1)\text{PremiumCustomer}(x) \rightarrow \text{PremiumCustomer}(x)$

R2 $\text{Quote}(x, y), \text{PremiumCustomer}(x), \text{LuxuryProduct}(y) \rightarrow \text{Discount}(x, y, 7.5\%)$

Rule **R0** states that if a quote is required for customer x , ask agent 2 whether x is a premium customer. **R1** states that agent 1 should believe agent 2 if agent 2 tells agent 1 that x is a premium customer, and **R2** states that premium customers get at 7.5% discount. Agent 2 is in charge of customer data, and has rules to determine who counts as a premium customer:

R3 $\text{Spending}(x, \text{min}5000, 2005) \rightarrow \text{PremiumCustomer}(x)$

R4 $\text{ask}(1, 2)\text{PremiumCustomer}(x), \text{PremiumCustomer}(x) \rightarrow \text{tell}(2, 1)\text{PremiumCustomer}(x)$

Suppose we are interested in quality of service guarantees of the form: every request for a quote is answered within at most 4 timesteps. To simplify, we have reformulated this as a question of whether agent 1 believes that a particular customer qualifies for a discount in at most 4 steps, provided that the information on customer spending and luxury products is immediately available. This can be re-expressed as a problem of verifying the formula

Q $\Box^4 B_1 \text{Discount}(\text{Miller}, \text{Porsche}, 7.5\%)$

(where \Box^n stands for n nestings of \Box) in a state where agent 1 believes $\text{Quote}(\text{Miller}, \text{Porsche})$ and $\text{LuxuryProduct}(\text{Porsche})$, and agent 2 believes $\text{Spending}(\text{Miller}, \text{min}5000, 2005)$. This reformulation presupposes that no other queries are being processed by agents 1 and 2.

It is straightforward to verify this property using existing model checking techniques. As proof of concept, we converted the example multi-agent system above into a system specification for the Mocha model checker [6] and verified that the formula **Q** is true. The translation into *reactive modules*, the description language used by Mocha, is straightforward: ground literals in the agent’s working memory are represented as boolean *state variables*, rules are represented by *atoms* (which describe the initial condition and transition relation for a group of related variables), and agents by *modules* (collections of atoms specifying which state variables are visible from outside the module). The multi-agent system is then simply a parallel composition of agent modules (this translation assumes an ‘all rules at each cycle’ rule application strategy, but ‘one instance at a time’ is also possible). The translation of \mathcal{ML} formula **Q** expressing the property to be verified into the ATL specification language used by Mocha is similarly straightforward.

This example is extremely simplified, and gives the response time guarantee in terms of ‘inference steps’ rather than of units of time. However, if we know the number of beliefs an agent has and the rule-matching strategy it uses, we can produce a mapping from ‘steps’ to time durations, and verify quality of service guarantees such as ‘every request for a quote will be answered within n milliseconds’.

8 RELATED WORK

Our work relates both to the field of epistemic logics (and their use in verifying properties of agents) and to work on verifying properties of rule-based programs, such as bounded response time.

A considerable amount of work has been done in the area of model-checking multi-agent systems (see, e.g., [9, 7]). However, the

emphasis of this work is on correctness rather than the timing properties of agents, which is our primary interest. An exception is our recent joint work on automatic verification of space and time requirements for resource-bounded agents [3, 2]. However this work focuses on verifying properties of a single agent and does not consider the multi-agent case.

Another strand of related work is verifying temporal properties of agent systems, in particular real-time properties, where actions are assumed to take non-trivial time. In [22], Singh proposed a framework for modelling agent systems where actions have duration and can be executed in parallel. As far as we know, it was not applied to actions as inferences. Recent work in dynamic and temporal epistemic logic [23] has a similar motivation to our work (modelling knowledge change) but assumes that the agent's knowledge is deductively closed before and after the update. There are, however, approaches in epistemic logic which explicitly reflect time required for a certain inference. For example, step logics [13] study development of agents' belief sets over discrete linear time. In [16] Grant, Kraus and Perlin present a semi-decidable formalism for expressing agent knowledge and inference steps with explicit time increments, which is implemented in Prolog. Other related work [12, 1, 4] also describes epistemic logics where each inference step takes the agent into the next (or some future) moment in time.

There has also been considerable work on the execution properties of rule based systems, both in AI and in the active database community. Perhaps the most relevant is that of Chen and Cheng on predicting the response time of OPS5-style production systems. In [10], they show how to compute the response time of a rule based program in terms of the maximum number of rule firings and the maximum number of basic comparisons made by the Rete network. In [11], Cheng and Tsai describe a tool for detecting the worst-case response time of an OPS5 program by generating inputs which are guaranteed to force the system into worst-case behaviour, and timing the program with those inputs.

9 CONCLUSION

We have presented a sound, complete and decidable logic which describes how the beliefs of communicating agents which reason using rules evolve over time. This logic can be used to express and verify temporal properties of multi-agent systems such as 'if agent i asks agent j λ , agent j is guaranteed to reply within n inference cycles'. This is useful, for example, for verifying quality of service guarantees, which may have the form of 'each query is going to be answered within n milliseconds'.

We are aware of a number of limitations of the work presented here. We currently assume that the agent's reasoning is monotonic (they never discard beliefs, only acquire new ones). It is, however, possible to expand our framework to incorporate rules which delete information, either to restore consistency or due to memory limitations and we plan to investigate this in future work. (An example of a similar logic which models a reasoner who deletes formulas to save memory can be found in [3].) Finally, the assumptions on communications are only made for the sake of simplicity; it is very easy to modify the semantics to allow for communication delays and lossy communication channels.

ACKNOWLEDGEMENTS

This work was partially supported by the Royal Society grant 'Model-checking resource-bounded agents'.

REFERENCES

- [1] T. Ågotnes and M. Walicki, 'Strongly complete axiomatizations of "knowing at most" in standard syntactic assignments', in *Proceedings of the Sixth International Workshop on Computational Logic in Multi-agent Systems (CLIMA VI)*, London, UK, (June 2005).
- [2] A. Allore, N. Alechina, P. Bertoli, C. Ghidini, B. Logan, and L. Serafini, 'Model-checking memory requirements of resource-bounded reasoners', in *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI'06)*. AAAI Press, (2006). (to appear).
- [3] N. Alechina, P. Bertoli, C. Ghidini, M. Jago, B. Logan, and L. Serafini, 'Verifying space and time requirements for resource-bounded agents', Technical Report T05-10-03, ITC-irst, Trento, Italy, (2005).
- [4] N. Alechina, B. Logan, and M. Whitsey, 'A complete and decidable logic for resource-bounded agents', in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pp. 606–613, New York, (July 2004). ACM Press.
- [5] N. Alechina, B. Logan, and M. Whitsey, 'Modelling communicating agents in timed reasoning logics', in *Proceedings of the Ninth European Conference on Logics in Artificial Intelligence (JELIA 2004)*, LNAI Vol. 3229, pp. 95–107, Lisbon, (September 2004). Springer.
- [6] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran, 'MOCHA: Modularity in model checking', in *Computer Aided Verification*, pp. 521–525, (1998).
- [7] M. Benerecetti, F. Giunchiglia, and L. Serafini, 'Model checking multiagent systems.', *J. Log. Comput.*, **8**(3), 401–423, (1998).
- [8] H. Boley, B. Grosop, and S. Tabet. RuleML tutorial, 2005. <http://www.ruleml.org/papers/tutorial-ruleml.html>.
- [9] R. Bordini, M. Fisher, W. Visser, and M. Wooldridge, 'State-space reduction techniques in agent verification', in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*, pp. 896–903, New York, (2004). ACM Press.
- [10] J.-R. Chen and A. M. K. Cheng, 'Predicting the response time of OPS5-style production systems', in *Proceedings of the 11th Conference on Artificial Intelligence for Applications*, p. 203. IEEE Computer Society, (1995).
- [11] A. M. K. Cheng and H. yen Tsai, 'A graph-based approach for timing analysis and refinement of OPS5 knowledge-based systems', *IEEE Transactions on Knowledge and Data Engineering*, **16**(2), 271–288, (2004).
- [12] H. N. Duc, 'Reasoning about rational, but not logically omniscient, agents', *Journal of Logic and Computation*, **7**(5), 633–648, (1997).
- [13] J. Elgot-Drapkin, M. Miller, and D. Perlin, 'Memory, reason and time: the Step-Logic approach', in *Philosophy and AI: Essays at the Interface*, 79–103, MIT Press, Cambridge, Mass., (1991).
- [14] E. Friedman-Hill, *Jess in Action: Java Rule-Based Systems*, Manning Publications Co., 2003.
- [15] M. P. Georgeff and A. L. Lansky, 'Reactive reasoning and planning', in *Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI-87*, pp. 677–682, (1987).
- [16] J. Grant, S. Kraus, and D. Perlin, 'A logic for characterizing multiple bounded agents', *Autonomous Agents and Multi-Agent Systems*, **3**(4), 351–387, (2000).
- [17] I. Horrocks and P. F. Patel-Schneider, 'A proposal for an OWL rules language.', in *Proceedings of the 13th international conference on World Wide Web, WWW 2004*, pp. 723–731. ACM, (2004).
- [18] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language, 2003. <http://www.daml.org/rules/proposal>.
- [19] Q. H. Mahmoud. Getting started with the Java rule engine API (JSP 94): Toward rule-based applications, 2005. <http://>
- [20] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. P. Sycara, 'The DAML-S virtual machine.', in *International Semantic Web Conference, LNCS Vol. 2870*, pp. 290–305. Springer, (2003).
- [21] M. Paolucci and K. Sycara, 'Autonomous semantic web services', *IEEE Internet Computing*, **7**(5), 34–41, (2003).
- [22] M. P. Singh, 'Toward a model theory of actions: How agents do it in branching time', *Computational Intelligence*, **14**(3), 287–305, (1998).
- [23] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi, 'Dynamic epistemic logic with assignment', in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)*, pp. 141–148. ACM New York, (2005).

Causation as Production

John Bell ¹

Abstract.

Recently I suggested that a cause is an event which, in its context of occurrence, is both necessary and sufficient for the effect. However this definition is only appropriate if there is a single potential cause of the effect. Consequently I suggest a generalization of the definition and discuss the resulting “Production Theory”. I suggest that this can be seen as a combination of a regularity theory in the Hume tradition and a dependence theory in the Lewis tradition, and argue that the Production Theory inherits the strengths of the component theories while avoiding their weaknesses.

1 INTRODUCTION

In [1] I propose a formal, logico-pragmatic, theory of causation. This defines a cause to be an event which, in its context of occurrence, is both necessary and sufficient for the effect. However I now consider that this definition is only appropriate in cases of non-redundant causation, where there is only one potential cause for the effect. Consequently I suggest an appropriate generalization, by weakening the contextual necessity condition, and justify the resulting “Production Theory” by relating it to the philosophical tradition and to recent philosophical discussions of causation.

The central theme of the discussion is Hall’s [8, p. 225] suggestion that causation should be viewed as production: that event c is a cause of effect e if c helps to *generate* or *bring about* or *produce* e . However, Hall argues that production should be analyzed in terms of regularity and that it conflicts with dependence, whereas I argue that production involves a combination of (appropriate forms of) regularity and dependence.

In §2 I briefly recall the regularity theories of Hume and Mill, provide an informal introduction to the theory of sufficient causation (the contextual-sufficiency part of the Production Theory) in this setting, and discuss the major problem confronting regularity theories; that of distinguishing between regularities which are causal and regularities which are accidental.

In §3 I recall Lewis’s theory of causation as counterfactual dependence and argue that, like the contextual necessity condition that I gave in [1], the simple form of dependence he adopts is too restrictive because it excludes symmetrically redundant causes. Consequently I suggest a weaker form counterfactual dependence called *individual dependence*. Then, in the resulting Production Theory, a cause is defined to be an event which, in its context of occurrence, is both sufficient and individually necessary for the effect. The Production Theory can thus be regarded as a combination of a regularity theory and a dependence theory.

In §4 I discuss Hall’s [7] argument that transitivity and dependence conflict, and argue that they do so only if, as in Lewis’s theory, depen-

dence is taken to be a sufficient condition for causation. I conclude that, contrary to Hall’s [p. 212] suggestion, dependence does “bear a deep connection to causation”, but it does so in conjunction with the theory of sufficient causation.

Finally, in §5 I discuss the varieties of redundancy, and argue that, while these cause problems for simple dependence theories, they do not do so for the Production Theory.

2 REGULARITY

Hume [10, Bk I, Pt III] suggests that we inductively acquire knowledge of regularities of succession of the form: A -type events are followed by B -type events. We then consider that A -type events cause B -type events, because whenever we see an A -type event, we expect it to be followed by a B -type event.

Mill [15, Bk III, Ch 5] complicates this picture by considering assemblages of conditions. A single assemblage might consist of an A -type event together with certain conditions which must be present (positive conditions) and certain conditions which must be absent (negative conditions). For example, an assemblage concerning the lighting of matches might include the striking of the match, the presence of oxygen, and the absence of dampness in the match head. This idea is complicated by the possibility of a plurality of causes; the possibility that an effect can be realized in more than one way. For example, a match can also be lit (in the presence of oxygen and the absence of dampness, etc.) by touching its head with a red hot poker. So assemblages should be thought of as disjunctions of conjunctions of conditions. Mill suggests that it is possible, at least in principle, to specify assemblages which, if true, logically guarantee their associated effects. However, this does not appear to be a realistic possibility because in practice it is impossible to come up with an exhaustive list, especially of negative conditions. There is also the problem of interference from other events.

Rather than begin with the concept of causation, I begin with a theory of events [4], in the AI reasoning-about-actions tradition [14, 16, 17, 18, 19], and use this as the basis for a definition of sufficient causation. A simplified version of the theory is described here. The full theory allows for the representation of context-dependent effects and non-deterministic effects; and hence of non-deterministic causation.

The theory is interpreted in a possible partial worlds setting. Each possible partial world (henceforth simply “world”) represents a possible history. At each world, time is taken to consist of points and to be discrete and linear. At any time point at a world, certain *facts* and certain *event occurrences* may be defined (be established as true or established as false). The idea behind the fact-event distinction is that events are dynamic (active) and introduce changes (impinge on the world, change the course of history), and facts are static (passive) and persist through time until affected by some event (until some event impinges on them). The partiality of worlds reflects notions such as

¹ Department of Computer Science, Queen Mary, University of London, United Kingdom, email: jb@dcs.qmul.ac.uk

partial observability (we typically do not have complete knowledge of a world) and relevance/localness (typically we are concerned only with a small region of it).

These ideas can be conveyed graphically by means of the Lewis-type neuron diagrams in Figure 1; reproduced from [8, Fig. 9.2]. In

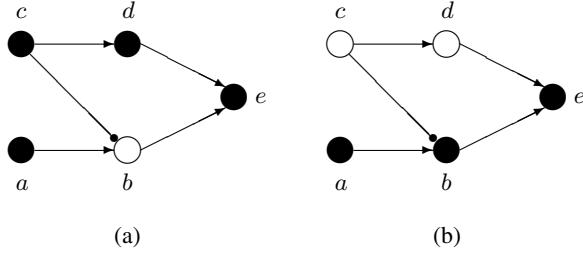


Figure 1. Two neuron diagrams: (a) c fires, (b) if c had not fired

these diagrams, shaded vertices represent firing neurons, arrows represent stimulatory connections, and the remaining edges represent inhibitory connections. Thus, in (b) the effect of a firing is the subsequent firing of b . However, in (a), this effect is inhibited by the firing of c . Figures (a) and (b) can be thought of as representing alternative world histories in which different neurons fire in the same neuron network.

World histories are described by the event language \mathcal{EL} . For example, the occurs atom $Occ(C)(1)$ states that neuron c fires at time 1 (at the world in question, at the actual world). The partial histories of worlds (a) and (b) can be represented by the following sets of atoms:

$$\begin{aligned} w_a &= \{Occ(C)(1), Occ(A)(1), Occ(D)(2), Occ(E)(3)\}, \\ w_b &= \{Occ(A)(1), Occ(B)(2), Occ(E)(3)\}. \end{aligned}$$

As non-firings have no effects they are, in the interests of economy, regarded as undefined occurrence atoms.

The behaviour of the neurons can be defined by regarding neuron firings as event types, and by defining their preconditions, effects, and interactions. This is done by the axioms in Table 1. For example,

Table 1. Sample event definition axioms.

$$\begin{aligned} \forall t(Pre(C)(t) &\equiv Occ(C)(t)) \\ &\wedge \forall t(Eff(C)(t) \equiv (Occ(D)(t) \wedge \neg T Occ(B)(t))) \quad (1) \\ \forall t(Pre(A)(t) &\equiv Occ(A)(t)) \\ &\wedge \forall t(Eff(A)(t) \equiv Occ(B)(t)) \wedge \forall t(Pre(C, A)(t) \\ &\equiv \neg Pre(A)(t)) \quad (2) \\ \forall t(Pre(D)(t) &\equiv Occ(D)(t)) \wedge \forall t(Eff(D)(t) \equiv Occ(E)(t)) \quad (3) \\ \forall t(Pre(B)(t) &\equiv Occ(B)(t)) \wedge \forall t(Eff(B)(t) \equiv Occ(E)(t)) \quad (4) \end{aligned}$$

Axiom (1) states that the precondition for c 's firing is that c occurs (in effect no preconditions are required in this case), and its effects are that d fires and it is not the case that b fires. The operator T , used in this axiom, is a classically-valued (truth-value designation) operator: a sentence $T\phi$ is true if ϕ is true, and is false otherwise (if ϕ is undefined or false). Consequently the combination $\neg T$ represents weak negation: the sentence $\neg T\phi$ is true if ϕ is false (if $\neg\phi$ is true) or ϕ is undefined, and is false otherwise. The truth operator makes it possible to reason classically with partial information; in particular, when using weak negation only two cases need to be considered; either sentence ϕ is true or it is not. The final conjunct of Axiom (2) is explained below.

Thus preconditions and effects embody specific causal knowledge. Their use is common in AI theories and has its origin in the repre-

sentation used by the planner STRIPS [6]. Note the similarities with Hume's regularities and Mill's assemblages.

Unlike most AI theories, the theory of events begins with the recognition that events are defeasible; that their preconditions are not always sufficient on occurrence for their effects. This may be because some unusual condition holds (the qualification problem), or because some other simultaneously occurring event interferes (the interaction problem), or, as Hume suggests, because we are mistaken in thinking that the regularity invariably holds (the problem of induction). In order to represent this, we begin with the notion of success. As Axiom (5) in Table 2 states, an event (token) succeeds (at time t) iff it is true that: the event occurs, its preconditions obtain, and its expected effects do indeed follow. The intention is that events

Table 2. The Production Theory of causation.

$$\forall e, t(Succ(e)(t) \equiv T(Occ(e)(t) \wedge Pre(e)(t) \wedge Eff(e)(t+1))) \quad (5)$$

$$\forall R, \bar{x}, t(Inert(R, \bar{x})(t) \equiv (R(\bar{x})(t) \equiv R(\bar{x})(t+1))) \quad (6)$$

$$\begin{aligned} \forall \epsilon, \phi, e, t(SCause(\epsilon, \phi) &\equiv \\ (\epsilon = Occ(e)(t) \wedge Succ(e)(t)) &\wedge \square((\theta \wedge Eff(e)(t+1)) \rightarrow \phi) \wedge \neg \square(\theta \rightarrow \phi))) \quad (7) \\ \forall \epsilon, \phi(Cause_1(\epsilon, \phi) &\equiv \\ (SCause(\epsilon, \phi) \wedge (\neg T\epsilon \Downarrow (\neg T\phi \vee \exists \epsilon' Cause_1(\epsilon', \phi)))))) \quad (8) \\ \forall \epsilon, \epsilon', \phi, \psi, \chi(Cause(\epsilon, \phi) &\equiv \\ (Cause_1(\epsilon, \phi) \vee (Cause_1(\epsilon, \epsilon') \wedge Cause(\epsilon', \phi))) \\ \vee (\phi = (\psi \wedge \chi) \wedge Cause(\epsilon, \psi) \wedge Cause(\epsilon, \chi)))) \quad (9) \end{aligned}$$

should normally succeed given their preconditions. So the preconditions should normally be sufficient on occurrence for the effect, and Axiom (5) should be used to infer by default that events succeed. So, given that the event occurs and its preconditions are true, $Occ(e)(t) \wedge Pre(e)(t)$, the *success assumption*, $Succ(e)(t)$, should, whenever consistent, be assumed, and the axiom used to infer its effects, $Eff(e)(t+1)$.

We can now return to the final conjunct of Axiom (2) which states that, in case of conflict, the success of event C is normally preferred to that of event A . Thus if, as in Figure 1(a), neurons c and a fire simultaneously, then the firing of c succeeds and the firing of a fails. More generally, event preferences can be used to represent the expected resolution of conflicts between events.

Axiom schema (6) defines inertia, and states that the fact represented by relation R and variables \bar{x} is inert at time t iff $R(\bar{x})(t)$ and $R(\bar{x})(t+1)$ have the same truth value. The axiom schema is intended to be used inductively to infer persistence whenever it is consistent to do so.

The *theory of events* consists of axioms (5) and (6) given in Table 2, and an *event theory* is a theory containing these axioms. The intended interpretation of event theories is enforced by their formal pragmatics (their nonmonotonic "semantics") [3], which interprets event theories chronologically, giving priority to success assumptions over inertia assumptions in case of conflict and respecting the event preferences of the theory.

The theory of causation is expressed in the more inclusive language \mathcal{CL} ; which admits quantification over \mathcal{EL} sentences and which includes the modal operators \square and \Downarrow , which refer across worlds.

Axiom (7) defines the notion of sufficient causation. It assumes a theory of background laws, which are represented as the conjunction θ and which include those defining the preconditions and effects of events. The axiom states that the occurrence of event e at time t is a *sufficient cause* (an *SCause*) of effect ϕ iff e succeeds at t , and ϕ is a logical consequence of the back-

ground laws together with e 's defined effects but not of the background laws alone. For example, $SCause(Occ(C)(1), Occ(D)(2))$ and $SCause(Occ(C)(1), \neg\text{TOcc}(B)(2))$ are true at w_a .

The theory of sufficient causation can be viewed as a regularity theory in the Humeian tradition. It avoids the problems of traditional regularity theories raised by Lewis [11, pp. 159-160]; in particular, the problem of preemption is discussed in §5. However, the theory suffers from the fundamental problem of regularity theories, that of distinguishing between those regularities of succession which represent genuine causal connections and those which are merely accidental. In particular, the theory is faced with the *inertia problem*, as it is unable to distinguish between effects which are produced by events and those which remain true by inertia; for example [1, p. 2], whitewashing a white wall is sufficient, but not necessary, for the wall being white. The theory thus fails to capture an important feature of causes, namely that they are events which are efficacious, which make a difference, which alter the course of history, which impinge, which "biff" (to use a technical term of Lewis's [13, pp. 283-4] non-technically).

3 DEPENDENCE

Lewis [11, Ch. 21] goes further and equates causation with difference: "We think of a cause as something that makes a difference, and the difference it makes must be a difference from what would have happened without it. Had it been absent, its effects—some of them, at least, and usually all—would have been absent as well" [pp. 160-1]. Consequently Lewis suggests that causation should be understood in terms of *counterfactual dependence* between pairs of events. Thus, for distinct events c and e , the occurrence of e is counterfactually dependent on the occurrence of c iff (1) c and e both occur, and (2) if c had not occurred, then e would not have occurred. Dependence of this kind will be called *simple dependence*. The causation relation is then taken to be the ancestral (transitive closure) of the (counterfactual) dependence relation.

Simple dependence can be represented in \mathcal{CL} by a conditional of the form $\neg\text{T}\phi \Downarrow \psi$, which, at a world at which ϕ is true, can, for present purposes, be understood as stating that ψ is true at the closest world at which $\neg\text{T}\phi$ is true; that removing ϕ from the world results in a world at which ψ is true. The formal pragmatics [3] ensures that the closest world to a given world is the expected one. (In general, there may be more than one closest world to a given world.) Thus, in the neuron example, both $\neg\text{TOcc}(C)(1) \Downarrow \neg\text{TOcc}(D)(2)$ and $\neg\text{TOcc}(C)(1) \Downarrow \text{Occ}(B)(2)$ are true at w_a (on the intended interpretation); as removing $\text{Occ}(C)(1)$ from w_a results in w_b . Consequently, on the basis of simple dependence, $\text{Occ}(C)(1)$ counts as the cause of both $\text{Occ}(D)(2)$ and $\neg\text{TOcc}(B)(2)$ at w_a .

Hall argues that simple dependence is inadequate as a definition of causation, because it is neither a necessary [8, §§3.2-3] nor a sufficient condition [7] for it. His argument against its sufficiency is discussed in the next section. His argument against its necessity can, I suggest, be made by appealing to cases of symmetrical redundancy.

Redundancy (a phenomenon discussed more generally in §5) arises when two (or more) separate potential causes for a certain effect are present and either one by itself would have been followed by the effect, and so the question arises as to which, if any, of the events the effect depends on. The problem is particularly acute when the redundancy is symmetrical, when both candidates have equal claim to being called causes of the effect; as nothing, either obvious or hidden, breaks the tie between them. Lewis [12, p. 80] suggests that our intuitions in these cases are unclear. So that we may consider that

neither event is a cause or consider that both events are causes; although we cannot reasonably say that one of the events is a cause and the other is not.

The institution of execution by firing squad exploits this uncertainty. Supposing that the riflemen in the squad all aim true and fire simultaneously, then the victim's death is redundantly caused, and so each rifleman is tempted to think that his shot was not the fatal one. On Lewis's simple dependence theory there is no cause in such cases; because the effect does not depend on any one of the candidate causes.

However, it seems more appropriate to say that each of the shots was a cause of the victim's death. This intuition is supported when causation is viewed as production. Each of the shots produces a non-inertial effect (biffs), so it seems odd to say that the effect is uncaused simply because there is a surplus of (candidate) causes. Overproduction may remove simple dependence, but it is, nevertheless, still production. This accounts for the guilt which the members of the firing squad feel; each knows that his shot was sufficient to kill the victim and that it would have done so even if all of the others had missed.

Consequently I suggest that simple dependence should be replaced by *individual dependence* as defined by Axiom (8) in Table 2. This states that event ϵ is a direct cause (a $Cause_1$) of effect ϕ iff ϵ is a sufficient cause of ϕ at a world and removing ϵ from the world results in a world at which either $\neg\text{T}\phi$ is true or some other event ϵ' directly causes ϕ . The individual dependence condition thus ensures that every $SCause$ in a given context biffs, but does not require that any of them biffs uniquely.

For example, in the case of the neurons, $\text{Occ}(C)(1)$ still counts as a cause of both $\text{Occ}(D)(2)$ and $\neg\text{TOcc}(B)(2)$, by sufficiency and simple dependence, at w_a . Moreover, we can obtain a formal analogue of the firing-squad example by supposing that (b) is changed so that d also fires, giving:

$$w_c = \{\text{Occ}(A)(1), \text{Occ}(D)(2), \text{Occ}(B)(2), \text{Occ}(E)(3)\}.$$

Then $Cause_1(\text{Occ}(D)(2), \text{Occ}(E)(3))$ and $Cause_1(\text{Occ}(B)(2), \text{Occ}(E)(3))$ are both true at w_c ; as removing either cause results in a world at which the other causes e to fire (by sufficiency and simple dependence).

The definition of $Cause_1$ is a generalization the definition of direct causation that I gave in [1]; which was restricted to cases of non-redundant or *singular* causation, and which can now be defined as follows: $\forall \epsilon, \phi (Cause_1(\epsilon, \phi) \equiv (Cause_1(\epsilon, \phi) \wedge \neg\exists \epsilon' (\neg\epsilon' = \epsilon \wedge SCause(\epsilon', \phi))))$.

4 TRANSITIVITY

Transitivity is a property that is naturally associated with production when we think of processes or causal chains; for example, in Figure 1(a) it seems natural to conclude that the firing of c (indirectly) causes the firing of e . Accordingly, as stated by Axiom (9) in Table 2, the (indirect) causation relation $Cause$ is taken to be transitive and to be closed under conjunction of effects. In this section three kinds of counterexample to transitivity are discussed.

Counterexamples of the first kind raise what might be called the *remoteness* objection. It seems odd to say that Queen Victoria's birth caused her death. However, according to the Production Theory, the statement is true; Queen Victoria's birth initiated a causal process leading ultimately to the events which directly caused her death. The oddness of the claim arises because, while Queen Victoria's birth was indeed an indirect cause of her death, her birth is not adequate

as an *explanation* of her death. An adequate explanation would consist of causes which were more proximate to her death, and which contributed more directly to it. But determining which causes constitute an adequate explanation is a pragmatic problem, not a metaphysical one. The metaphysical task is to define the causal chain, the pragmatic task is to decide which parts of the chain are relevant to a particular explanation.

An example of the second kind is Hall's [7, p. 187] engineer who: "is standing by a switch in the railroad tracks. A train approaches in the distance. She flips the switch, so the train travels down the right-hand track instead of the left. Since the tracks reconverge up ahead, the train arrives at its destination all the same; let us further suppose that the time and manner of its arrival are exactly as they would have been had she not flipped the switch. Let c be the engineer's action and e be the train's arrival. Pick an event d that is part of the train's journey down the right-hand track. Clearly c is a cause of d and d of e ; but is c a cause of e ? Is her flipping the switch a cause of the train's arrival?" Intuitively it might seem not, because it seems to be clear that the switching event makes no difference to whether the train arrives, but merely determines the *route* by which it arrives. However, Hall [§§4-5] argues that, despite this consideration, flipping the switch is a cause of the train's arrival. If, as the example persuades us to do, we think in terms of the train's arrival and *dependence*, then we are inclined to think that if the switch had not been flipped, the train would have arrived just the same, and so the switch cannot be considered to be a cause. However, if we think in terms of how the train's arrival was *produced*, then we need to consider the causal chain leading to it, and clearly this includes the switching event. In Hall's terminology, the switching event is a "causal switch", as it changes the course of events; without it a different sequence of events would have produced (caused) the train's arrival. Readers who are unconvinced by this summary are referred to Hall's detailed and persuasive discussion.

Formally, suppose an event theory analogous to the one for the neurons, which determines the following partial histories:

$$\begin{aligned} w_0 &= \{Occ(C)(1), Occ(D)(2), Occ(E)(3)\}, \\ w_1 &= \{Occ(C)(1)\}, \\ w_2 &= \{Occ(B)(2), Occ(E)(3)\}. \end{aligned}$$

Thus w_0 represents the actual world in which the switch is flipped, and the train travels via the right-hand track; w_1 represents the world where the switch is flipped, but the train's progress on the right-hand track is obstructed; and w_2 represents the world in which the switch is not flipped, and the train travels down the left-hand track. Then both $Cause_1(Occ(C)(1), Occ(D)(2))$ and $Cause_1(Occ(D)(2), Occ(E)(3))$ are true at w_0 . Consequently $Cause(Occ(C)(1), Occ(E)(3))$ is true at w_0 by Axiom (9).

Examples of the third kind involve cases of what Hall calls "double prevention", and are designed to show that simple dependence cannot be a sufficient condition for causation if causation is taken to be transitive. In one of his examples [pp. 183-4]: "Billy sees Suzy about to throw a water balloon at her neighbor's dog. He runs to try to stop her, but trips over a root and so fails. Suzy, totally oblivious to him, throws the water balloon at the dog. . . . Billy's running toward Suzy causes him to trip, which in turn causes the dog to yelp (by *Dependence*: if he hadn't tripped, he would have stopped her from throwing and so the dog wouldn't have yelped). But, intuitively, his running toward her does not cause the dog's yelp". In this example, Billy's trip is a double preventer because it prevents him from preventing her from throwing the balloon. Hall rightly suggests that

double preventers are not causes and concludes [p. 182] that: "the price of transitivity—a price well worth paying—is to give up the claim that there is any *deep* connection between counterfactual dependence and (the central kind of) causation". As there is no distinction between simple and individual dependence in examples like the one given, it seems that they challenge the necessity of individual dependence. However Hall's double-prevention examples all assume that dependence is a sufficient condition for causation (note, for instance, the appeal to dependence in the *Billy-Suzy* example), and so do not challenge the Production Theory because dependence is not taken to be a sufficient condition for causation; any $Cause_1$ is also required to be an $SCause$.

5 REDUNDANCY

Symmetrical redundancy is discussed in §3, and it arises when two or more potential causes have equal claim to being regarded as the cause of an effect. Other redundancies are asymmetrical: intuition suggests that one of the events is a cause and the other is not; that one of the events is a *preempting* cause and that the other is a *preempted* cause. Lewis [12, p. 80] comments that: "When our opinions are clear, it's incumbent on an analysis of causation to get them right. This turns out to be a severe test".

Figure 1(a) provides an example of what Lewis calls *early pre-emption*. Both a and c fire simultaneously, with the firing of c causing (and thereby preempting the firing of a from causing) the firing of e . On Lewis's theory c counts as the cause because of the stepwise dependence of the firing of e on the firing of d , and the firing of d on the firing of c . In the case of the Production Theory, suppose the following partial histories:

$$\begin{aligned} w_0 &= \{Occ(C)(1), Occ(A)(1), Occ(D)(2), Occ(E)(3)\}, \\ w_1 &= \{Occ(C)(1), Occ(A)(1)\}, \\ w_2 &= \{Occ(A)(1), Occ(B)(2), Occ(C)(3)\}. \end{aligned}$$

Here w_0 represents the actual world, w_1 represents the alternative history in which the occurrence of D is removed, and w_2 represents the alternative history in which the occurrence of C is removed. Then $\neg TOcc(C)(1) \Downarrow \neg TOcc(D)(2)$ and $\neg TOcc(D)(2) \Downarrow \neg TOcc(E)(3)$ are both true at w_0 . So, $Cause_1(Occ(C)(1), Occ(D)(2))$ and $Cause_1(Occ(D)(2), Occ(E)(3))$ are true at w_0 . Consequently $Cause(Occ(C)(1), Occ(E)(3))$ is true at w_0 (by Axiom (9)).

The most common examples of asymmetrical redundancy are cases of what Lewis calls *late pre-emption*. In such cases the preempting cause produces the effect before the preempted alternative can do so. Wright [20] gives the example of two forest fires, A and B approaching a house from opposite directions. Suppose that A arrives first and destroys the house. Then it seems reasonable to conclude that A caused the destruction; despite the fact that if A not done so, then B would have. Halpern and Pearl [9, p. 1] motivate their theory of causation as contingent dependency with this example. They suggest that the house burning down depends on (is caused by) fire A under the contingency that the fire fighters arrive and extinguish fire B before it reaches the house, but not otherwise. This seems wrong. For example, if A was started by arsonist a , B was started by rival arsonist b , and B was not extinguished by the fire fighters, then a would be considered to be guilty of arson, and b of attempted arson. The usual, and it seems correct, solution to cases of this kind is to appeal to the time at which events occur. This is done in the formal treatment of the example in [1, pp. 9-10], A destroys the house at

time t thereby preempting B from doing so at the later time t' as the house is no longer flammable at t' . A preempts B by removing one of B 's preconditions, and so the conflict is resolved at the "event level" (by the event theory).

However in cases of *trumping preemption* this resource is not available, as the two candidate causes are followed by the very same effect. Lewis [12] gives the following example, which was suggested by van Fraassen. Suppose that a sergeant and a major simultaneously order their soldiers to advance and that the soldiers do so. Their advance is redundantly caused, since either order would, on its own, have been sufficient. However, the redundancy is asymmetric, since the soldiers obey the senior officer. The soldiers advance because the major orders them to, not because the sergeant does. The major's order trumps the sergeant's. Once again, the formal treatment [pp. 10–11] shows that an event-level resolution is possible by making it a precondition of issuing an order that no more senior soldier issues an order at the same time; with the consequence that the major's order succeeds and the sergeant's fails. An alternative, more flexible, solution is to use an event preference to represent the fact that the senior officer's orders normally override conflicting orders from the junior officer.

Redundancy causes difficulty for dependence considered as a sufficient condition. Indeed, the problem of trumping prompted Lewis to abandon simple dependence in favour of the more complex and vague notion of counterfactual covariance [12]. However, it does not appear to pose problems for the Production Theory because conflicts can be resolved at the event level (by the underlying event theory). This seems to be the appropriate place because the resolution of conflicts depends on specific causal knowledge of the kind encoded in the definitions of events and event preferences. This also has the advantage of making it possible to give a simple, general, definition of causation.

6 CONCLUDING REMARKS

I agree with Hall that causation as production should be seen as the central metaphysical notion of causation, and suggest that this notion can be formalized by combining appropriate forms of regularity and dependence. The resulting Production Theory captures two significant features of production: regular succession (expected/normally dependable outcome) and making a difference in context (efficacy), and inherits the strengths of regularity and dependence theories while avoiding their weaknesses. The Production Theory also offers a clear division between a general, and relatively simple, definition of causation, and particular causal knowledge embodied in the definition of events and event preferences.

As the common sense concept of causation is complex and vague, any comprehensible formal theory of it must inevitably be prescriptive rather than descriptive. In a related paper [2] I consider the more exotic causal notions of prevention and causation by absence, and show that prevention can be reduced to counterfactual production and that causation by absence can be reduced to counterfactual production together with a pragmatic parameter. For example, the ordinary language claim "Lack of food causes hunger" can be analyzed as the counterfactual 'If food were present, then eating would cause (produce) an absence of hunger' provided that it is assumed that eating normally occurs when food and hunger are present.

These considerations suggest that the Production Theory is of some philosophical interest. Moreover as the theory is entirely formal, it can be implemented (in simple cases at least) and used by artificial agents to reason about actual and counterfactual causation.

ACKNOWLEDGEMENTS

I am grateful to the reviewers and to everyone else who has commented on this work.

REFERENCES

- [1] John Bell, 'Causation and causal conditionals', in *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pp. 2–11. AAAI Press, (2004).
- [2] John Bell. Causation and absence, 2006. Available at: www.dcs.qmul.ac.uk/~jb.
- [3] John Bell. A common sense theory of causation, 2006. Available at: www.dcs.qmul.ac.uk/~jb.
- [4] John Bell. Natural events, 2006. Available at: www.dcs.qmul.ac.uk/~jb.
- [5] John Collins, Ned Hall, and L. A. Paul (Editors), *Causation and Counterfactuals*, MIT Press, Cambridge, Massachusetts, 2004.
- [6] R. Fikes and Nils J. Nilsson, 'STRIPS: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, **2**, 189–208, (1971).
- [7] Ned Hall, 'Causation and the price of transitivity', In Collins et al. [5], pp. 181–203.
- [8] Ned Hall, 'Two concepts of causation', In Collins et al. [5], pp. 225–276.
- [9] Joseph Halpern and Judea Pearl, 'Causes and explanations: A structural model approach. Part I: Causes', in *Proceedings of the Seventeenth Conference on Uncertainty in AI (UAI 2001)*, pp. 194–202, (2001). Extended version available at: www.cs.cornell.edu/home/halpern.
- [10] David Hume, *A Treatise of Human Nature*, Clarendon Press, Oxford, 1978. First published in 1739.
- [11] David Lewis, *Philosophical Papers*, volume II, Oxford University Press, Oxford, 1986.
- [12] David Lewis, 'Causation as influence', In Collins et al. [5], pp. 75–106.
- [13] David Lewis, 'Void and object', In Collins et al. [5], pp. 277–290.
- [14] John McCarthy, 'Applications of circumscription to formalizing commonsense knowledge', *Artificial Intelligence*, **28**, 89–116, (1986).
- [15] J.S. Mill, *A System of Logic*, Longmans, London, 1941. First published in 1898.
- [16] Raymond Reiter, *Knowledge in Action; Logical Foundations for Specifying and Implementing Dynamical Systems*, MIT Press, Cambridge, Mass., 2001.
- [17] Murray Shanahan, *Solving the Frame Problem; A Mathematical Investigation of the Common Sense Law of Inertia*, MIT Press, Cambridge, Mass., 1997.
- [18] Yoav Shoham, *Reasoning About Change*, MIT Press, Cambridge, Mass., 1988.
- [19] Yoav Shoham, 'Nonmonotonic reasoning and causation', *Cognitive Science*, **14**, 213–252, (1990).
- [20] R. Wright, 'Causation, responsibility, risk, probability, naked statistics and proof: Pruning the bramble bush by clarifying the concepts', *Iowa Law Review*, **73**, 1001–1077, (1988).

Merging possibilistic networks

Salem Benferhat¹

Abstract. This paper deals with merging multiple-source uncertain pieces of information, which are encoded by means of possibilistic networks. We first show that the merging of possibilistic networks having the same graphical structure can be easily achieved in polynomial time. When possibilistic networks have different graphical structures we show that their fusion can also be efficiently done by extending initial possibilistic networks into a same common structure. We then address two important problems: how to deal with cycles, and how to solve the subnormalization problem which reflects conflicts between sources?

1 Introduction

Possibilistic networks [11, 12] are important tools proposed for an efficient representation of uncertain information. Their success is due to their simplicity of representing and handling independence relationships. Possibilistic networks are directed acyclic graphs (DAG), where each node encodes a variable and every edge represents a relationship between two variables. Uncertainties are expressed by means of conditional possibility distributions for each node in the context of its parents.

This paper addresses the problem of merging uncertain pieces of information represented by possibilistic networks. The fusion of uncertain pieces of information issued from different sources is an important issue that can be encountered in various fields of applications.

The problem of fusing propositional or weighted logical knowledge bases, issued from different sources, was studied in several works [1, 6, 8, 3]. However, few works have been done for fusing possibilistic networks (e.g., [2]). These recent works are not satisfactory, since either they are computationally hard, or their use is very limited to special cases.

This paper proposes alternative approaches which overcome limitations of existing approaches. Its main contributions are:

- We first propose efficient ways for fusing possibilistic networks having the same graphical structure
- We show how to expand possibilistic networks (by adding new variables and/or links) without modifying joint distributions
- We then consider the case where possibilistic networks to fuse have different graphical structures. The idea is : i) to compute a common structure (which is basically the union of the initial graphs), ii) to expand each network to this common structure, and iii) to apply the fusion procedure

defined for merging possibilistic networks having the same graphical structure.

- We address the problem of cycles. Indeed, even if initial possibilistic networks are free of cycles, it may happen that their union contains cycles. We show that existing propagation algorithms, in particular the well-known junction tree algorithm, can be extended to the case with cycles.
- Lastly, we deal with the normalization problem which expresses the presence of conflicts between sources.

The rest of this paper is organised as follows. Next section gives a brief background on possibility theory. Section 3 presents the fusion of possibilistic networks. Section 4 deals with cycles and normalization problems. Section 5 concludes the paper.

2 Basics of possibility theory

Let $V = \{A_1, A_2, \dots, A_N\}$ be a set of variables. We denote by $D_A = \{a_1, \dots, a_n\}$ the domain associated with the variable A , and a denotes any instance of A . $\Omega = \times_{A_i \in V} D_{A_i}$ denotes the universe of discourse, which is the Cartesian product of all variable domains in V . In the following, we only give a brief overview on possibility theory, for more details see [10].

2.1 Possibility distributions

A possibility distribution π is a mapping from Ω to the interval $[0, 1]$. It represents a state of knowledge about a set of possible situations distinguishing what is plausible from what is less plausible. Given a possibility distribution π , we can define a mapping grading the *possibility measure* of an event $\phi \subseteq \Omega$ by $\Pi(\phi) = \max_{\omega \in \phi} \pi(\omega)$. A possibility distribution π is said to be *normalized*, if $\max_{\omega} \pi(\omega) = 1$.

In this paper, we only use the so-called min-based conditioning proposed in an ordinal setting [10] [13], and defined by:

$$\Pi(\psi | \phi) = \begin{cases} \Pi(\psi \wedge \phi) & \text{if } \Pi(\psi \wedge \phi) < \Pi(\phi) \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

2.2 Possibilistic networks revisited

In existing works, graphs associated with possibilistic networks are DAGs (Directed Acyclic Graph). However, as we will see later propagation algorithms can be adapted to the case of graphs with cycles. Therefore, we provide a revised definition of possibilistic networks, where we simply relax the condition that the associated graph should be a DAG.

¹ CRIL-CNRS, Université d'Artois, Rue Jean Souvraz 62307 Lens Cedex, France. Email: benferhat@cril.univ-artois.fr

Namely, a *min-based possibilistic network* or simply *possibilistic network* over a set of variables V , denoted by $\mathbb{N} = (\pi_{\mathbb{N}}, G_{\mathbb{N}})$, consists of:

- a *graph*, denoted by $G_{\mathbb{N}}$, where nodes represent variables and edges encode the link between the variables. The parent set of a node A is denoted by U_A .
- a *numerical component*, denoted by $\pi_{\mathbb{N}}$, which quantifies different links. For every root node A uncertainty is represented by the conditional possibility distributions $\pi_{\mathbb{N}}(A | u_A)$ over A in the context of each $u_A \in D_{U_A}$.

When $G_{\mathbb{N}}$ is a DAG, we call \mathbb{N} a standard possibilistic network. The set of a priori and conditional possibility degrees induces a unique joint possibility distribution defined by:

Definition 1 Given the a priori and conditional possibility distributions, the joint distribution denoted by $\pi_{\mathbb{N}}$, is expressed by the following min-based chain rule (which the counterpart of probabilistic chain rule) :

$$\pi_{\mathbb{N}}(A_1, \dots, A_N) = \min_{i=1 \dots N} \pi_{\mathbb{N}}(A_i | U_{A_i}) \quad (2)$$

3 Merging min-based possibilistic networks

3.1 Conjunctive merging

One of the important aims in merging uncertain information is to exploit complementarities between the sources in order to get a more complete and precise global point of view.

In possibility theory, given a set of possibility distributions π'_i 's, the basic combination mode is the conjunction (i.e., the minimum) of possibility distributions. Namely:

$$\forall \omega, \pi_{\oplus}(\omega) = \min_{i=1, \dots, n} \pi_i(\omega).$$

The conjunctive aggregation mode makes sense if all the sources are regarded as equally reliable (for more details see [5]).

3.2 The problem

Our aim is to directly construct from n standard possibilistic networks $\mathbb{N}_1, \dots, \mathbb{N}_n$, issued from different sources, a new possibilistic network, denoted by \mathbb{N}_{\oplus} .

The new possibilistic network should be such that:

$$\forall \omega, \pi_{\mathbb{N}_{\oplus}}(\omega) = \min_{i=1, \dots, n} \pi_{\mathbb{N}_i}(\omega),$$

where $\pi_{\mathbb{N}_{\oplus}}, \pi_{\mathbb{N}_i}$'s are respectively the possibility distributions associated with \mathbb{N}_{\oplus} and \mathbb{N}_i 's using Definition 1.

Recently, in [2] a preliminary work has been proposed to fuse possibilistic networks. Unfortunately, this work is not satisfactory. Indeed, it only considers a very restricted case where the union of graphs, associated with possibilistic networks to fuse, is free of cycles. This is a real limitation. For instance, assume that two sources have a same joint distribution π defined on a set of two variables $\{A, B\}$. Assume moreover, that these two sources use different possibilistic networks to represent this distribution. The first source defines a network where we have a link from A to B . While the other source uses the converse link from B to A . Since the union of the graphs contains a cycle, the approach proposed in [2] cannot

be applied, even if the possibility distribution, associated with possibilistic networks is the same.

Moreover, the approach proposed in [2] does not deal with the sub-normalization problem. Hence, this approach is not appropriate when the sources are conflicting.

Another way to deal with the fusion of possibilistic networks is to exploit the equivalent transformations between possibilistic graphs and possibilistic knowledge bases. Namely, to fuse possibilistic networks : i) first transform initial possibilistic networks into their associated possibilistic knowledge bases using results given in [4], ii) fuse the obtained knowledge bases using results of [3], and iii) transform the fused possibilistic knowledge base into a fused possibilistic network using results of [4].

This procedure has at least three main limitations. First, the transformation step from a possibilistic base into a possibilistic graph is generally a hard problem. Secondly, it does not preserve initial structure between variables. For instance, if the two graphs to fuse have exactly the same singly connected DAG (for which the propagation algorithm is efficient), then the result can be a multiply connected DAG (for which the propagation algorithm is hard). Lastly, existing transformations only deal with binary variables.

This paper proposes alternative approaches for fusing possibilistic networks that overcome the above limits. Let us start with the construction of \mathbb{N}_{\oplus} when the standard possibilistic networks \mathbb{N}_i 's have exactly the same DAG.

3.3 Fusion of the same-structure networks

This section presents the procedure of merging possibilistic networks of a same DAG structures. The possibilistic networks to merge, denoted by \mathbb{N}_i 's, only differ on conditional possibility distributions assigned to variables. The following definition and proposition show that the result of merging of networks is immediate.

Definition 2 Let $\mathbb{N}_i = (\pi_{\mathbb{N}_i}, G_{\mathbb{N}_i})$'s ($i = 1, \dots, n$) be n possibilistic networks such that :

$\forall i = 1, \dots, n, \forall j = 1, \dots, n$ we have $G_{\mathbb{N}_i} = G_{\mathbb{N}_j}$.

The result of merging \mathbb{N}_i 's is a possibilistic network denoted by $\mathbb{N}_{\oplus} = (\pi_{\mathbb{N}_{\oplus}}, G_{\mathbb{N}_{\oplus}})$, where :

- $G_{\mathbb{N}_{\oplus}} = G_{\mathbb{N}_i}$ (for any $i = 1, \dots, n$) and
- $\pi_{\mathbb{N}_{\oplus}}$ is defined by:

$$\forall A, \pi_{\mathbb{N}_{\oplus}}(A | U_A) = \min_{i=1, \dots, n} \pi_{\mathbb{N}_i}(A | U_A).$$

Proposition 1 Let $\mathbb{N}_i = (\pi_{\mathbb{N}_i}, G_{\mathbb{N}_i})$'s ($i = 1, \dots, n$) be n standard possibilistic networks having exactly the same associated DAG. Let $\mathbb{N}_{\oplus} = (\pi_{\mathbb{N}_{\oplus}}, G_{\mathbb{N}_{\oplus}})$ be the result of merging \mathbb{N}_i 's using the above definition. Then, we have :

$$\forall \omega, \pi_{\mathbb{N}_{\oplus}}(\omega) = \min_{i=1, \dots, n} \pi_{\mathbb{N}_i}(\omega),$$

where $\pi_{\mathbb{N}_{\oplus}}, \pi_{\mathbb{N}_i}$'s are respectively the possibility distributions associated with \mathbb{N}_{\oplus} and \mathbb{N}_i 's using Definition 1.

3.4 Fusion of different-structure networks

The above subsection has shown that the fusion of possibilistic networks can be easily achieved if they have the same DAG. This section considers the case when the networks have not

necessarily the same structure. The idea is to expand initial possibilistic networks into a common structure, which is simply the union of initial graphs.

A union of a set of DAGs (G_1, \dots, G_n) is a graph where : i) the set of its variables is the union of variables in G_1, \dots, G_n and ii) for each variable A, its parents are those in G_1, \dots, G_n .

The two following propositions are very useful since they show that any possibilistic network can be extended with additional variables and/or links. The following concerns the addition of variables.

Proposition 2 Let $\mathbb{N} = (\pi_{\mathbb{N}}, G_{\mathbb{N}})$ be a possibilistic network defined on a set variables V . Let A be a new variable. Let $\mathbb{N}^E = (\pi_{\mathbb{N}}^E, G_{\mathbb{N}}^E)$ be a new possibilistic networks such that :

- $G_{\mathbb{N}}^E$ is such that : i) the set of variables of $G_{\mathbb{N}}^E$ is the one of $G_{\mathbb{N}}$ plus the new variable A , and ii) the set of arcs between variables in $G_{\mathbb{N}}^E$ is exactly the same as the one in $G_{\mathbb{N}}$ (the new variable A has neither parents nor children).
- $\pi_{\mathbb{N}}^E$ is identical to $\pi_{\mathbb{N}}$ for variables in V , and is equal to a uniform possibility distribution on the node A (namely, $\forall a \in D_A, \pi_{\mathbb{N}}^E(a) = 1$).

Then, we have :

$\forall \omega \in \times_{A_i \in V} D_{A_i}, \pi_{\mathbb{N}}(\omega) = \max_{a \in D_A} \pi_{\mathbb{N}}^E(a\omega)$, where $\pi_{\mathbb{N}}$ and $\pi_{\mathbb{N}}^E$ are respectively the possibility distributions associated with \mathbb{N} and \mathbb{N}^E using Definition 1.

If \mathbb{N}^E is the obtained augmented network, then the possibility distribution $\pi_{\mathbb{N}}$ can be recovered by marginalizing $\pi_{\mathbb{N}}^E$ on the new added variables. We now provide a proposition which shows how to add links to a possibilistic network without changing its possibility distribution.

Proposition 3 Let $\mathbb{N} = (\pi_{\mathbb{N}}, G_{\mathbb{N}})$ be a possibilistic network. Let A be a variable. Let $B \notin Par(A)$. Let $\mathbb{N}^E = (\pi_{\mathbb{N}}^E, G_{\mathbb{N}}^E)$ be a new possibilistic network obtained from $\mathbb{N} = (\pi_{\mathbb{N}}, G_{\mathbb{N}})$ by adding a link from B to A . The new conditional possibility distribution associated with A is:

$$\forall a \in D_A, b \in D_B, u \in D_{Par(A)}, \pi_{\mathbb{N}}^E(a | ub) = \pi_{\mathbb{N}}(a | u).$$

Then, we have :

$$\forall \omega, \pi_{\mathbb{N}}(\omega) = \pi_{\mathbb{N}}^E(\omega).$$

Given these two above propositions the fusion of n possibilistic networks $\mathbb{N}_1, \dots, \mathbb{N}_n$ is immediate, using the following two steps:

Step 1 Let $G_{\mathbb{N}\oplus}$ be the union of $G_{\mathbb{N}1}, \dots, G_{\mathbb{N}n}$. Using Propositions 2 and 3, expand $\mathbb{N}_1, \dots, \mathbb{N}_n$ to $\mathbb{N}_1^E, \dots, \mathbb{N}_n^E$ such that $\forall i = 1, \dots, n, G_{\mathbb{N}_i}^E = G_{\mathbb{N}\oplus}$.

Step 2 As in Definition 2, define conditional possibility distribution $\pi_{\mathbb{N}\oplus}(A|U_A)$ associated with each variable given their parents, namely:

$$\forall A, \pi_{\mathbb{N}\oplus}(A|U_A) = \min_{i=1, \dots, n} \pi_{\mathbb{N}_i}^E(A|U_A).$$

Proposition 4 Let $\mathbb{N}\oplus$ be the result of merging \mathbb{N}_i 's using the above steps 1 and 2. Then, we have :

$$\forall \omega, \pi_{\mathbb{N}\oplus}(\omega) = \min_{i=1, \dots, n} \pi_{\mathbb{N}_i}(\omega).$$

Example 1 Let us consider two standard possibilistic networks, where their DAG are given by Figure 1. These two DAGs have a different structure. Figure 2 provides the DAG of $G_{\mathbb{N}\oplus}$. Note that $G_{\mathbb{N}\oplus}$ contains a cycle.

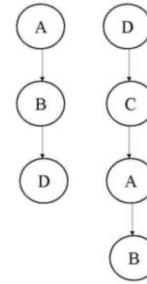


Figure 1. $G_{\mathbb{N}1}$ and $G_{\mathbb{N}2}$: Initial DAGs

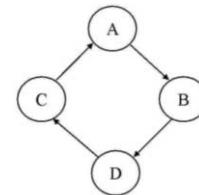


Figure 2. $G_{\mathbb{N}\oplus}$: Fused graph of $G_{\mathbb{N}1}$ and $G_{\mathbb{N}2}$

Tables 1-4 give conditional possibility distributions associated with $\mathbb{N}_1^E, \mathbb{N}_2^E$ and $\mathbb{N}\oplus$. Initial possibility distributions associated with \mathbb{N}_1 and \mathbb{N}_2 appear in bold columns. In our example, possibility distributions associated with expanded networks are either equal to initial possibility distributions or to uniform distributions.

From these different tables of conditional distributions, we can easily show that the joint possibility of $\pi_{\mathbb{N}\oplus}$ computed by chain rule, is equal to the minimum of $\pi_{\mathbb{N}1}$ and $\pi_{\mathbb{N}2}$. For instance, let $\omega = a_1 b_2 c_1 d_1 \in \Omega$. Using chain rule, we have:
 $\pi_{\mathbb{N}1}(a_1 b_2 c_1 d_1) = .4, \pi_{\mathbb{N}2}(a_1 b_2 c_1 d_1) = .8$
and $\pi_{\mathbb{N}\oplus}(a_1 b_2 c_1 d_1) = .4$.

Table 1. Possibility distributions associated with B given A

A	B	$\pi_{\mathbb{N}1}^E = \pi_{\mathbb{N}1}$	$\pi_{\mathbb{N}2}^E = \pi_{\mathbb{N}2}$	$\pi_{\mathbb{N}\oplus}$
a_1	b_1	.1	.8	.8
a_1	b_2	.4	1	.4
a_2	b_1	.6	1	.6
a_2	b_2	1	.7	.7

4 Dealing with cycles and subnormalized joint distributions

4.1 Dealing with cycles

Results of above sections guarantee that the joint distribution associated with the fused possibilistic networks is indeed equal to a conjunctive combination of joint distributions associated with initial standard possibilistic networks. However, even if initial graphs are free of cycles it may happen that their union produces cycles. This was illustrated by our running example.

Table 2. Possibility distributions associated with D given B , where – means that the link $B \rightarrow A$ does not exists in $G_{\text{N}2}$

B	D	$\pi_{\text{N}1} = \pi^E_{\text{N}1}$	$\pi_{\text{N}2}$	$\pi^E_{\text{N}2} = \pi_{\text{N}2}$	$\pi_{\text{N}\oplus}$
b_1	d_1	1	-	1	1
b_1	d_2	.9	-	1	.9
b_2	d_1	1	-	1	1
b_2	d_2	.8	-	1	.8

Table 3. Possibility distributions associated with C given D , where – means that the link $C \rightarrow D$ does not exists in $G_{\text{N}1}$

D	C	$\pi_{\text{N}1}$	$\pi^E_{\text{N}1}$	$\pi^E_{\text{N}2} = \pi_{\text{N}2}$	$\pi_{\text{N}\oplus}$
d_1	c_1	-	1	.9	.9
d_1	c_2	-	1	1	1
d_2	c_1	-	1	1	1
d_2	c_2	-	1	.4	.4

This section shows that, due to the idempotency of the conjunctive operator, the presence of cycles can be easily managed in a possibility framework. More precisely, we show that the propagation algorithm for multiply connected graphs can be easily adapted for dealing with cycles.

The idea is, as for multiply-connected probabilistic networks, to proceed to a transformation of the fused network $G_{\text{N}\oplus}$ into a junction tree structure. Given a cyclic fused possibilistic network $G_{\text{N}\oplus}$, the construction of its corresponding junction tree is performed exactly in the same manner as in the probabilistic case [7]. In a first step, the graph is moralized by adding undirected edges between the parents and by dropping the direction of existing edges. Then, the moral graph is triangulated. Finally, the triangulated graph is transformed into a junction tree where each node represents a *cluster* of variables and each edge is labelled with a *separator* corresponding to the intersection of its adjacent clusters.

Figure 3 gives the junction tree associated with the fused graph $G_{\text{N}\oplus}$ of figure 2 (there are two clusters $C_1 = \{ABC\}$ and $C_2 = \{BCD\}$ and one separator $\{BC\}$ which is the intersection of the two clusters).

Once the junction tree is constructed, the main steps of the junction tree propagation algorithm are :

- **Step 1 (Initialization) :**

- For each cluster $C_i : \pi_{C_i}^I \leftarrow \mathbf{1}$, where $\mathbf{1}$ is a possibility distribution where all elements have a highest possibility degree 1.
 - For each separator $S_{ij} : \pi_{S_{ij}}^I \leftarrow \mathbf{1}$,
 - For each variable A , select a cluster C_i containing $A \cup U_A$ (such cluster exists, thanks to the moralisation step) and update its possibility distributions as follows :
- $$\pi_{C_i}^I : \pi_{C_i}^I \leftarrow \min(\pi_{C_i}^I, \Pi(A | U_A)).$$

Proposition 5 Let $\text{N}\oplus$ be the result of merging N_i 's given in Proposition 4. Let $\pi_{C_i}^I$ be the possibility distribution

Table 4. Possibility distributions associated with A given C , where – means that the link $C \rightarrow A$ does not exists in $G_{\text{N}1}$

C	A	$\pi^E_{\text{N}1}$	$\pi_{\text{N}1}$	$\pi^E_{\text{N}2} = \pi_{\text{N}2}$	$\pi_{\text{N}\oplus}$
c_1	a_1	-	1	.8	.8
c_1	a_2	-	1	1	1
c_2	a_1	-	1	.4	.4
c_2	a_2	-	1	1	1

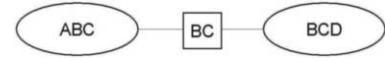


Figure 3. Junction tree associated with graph $G_{\text{N}\oplus}$ of figure 2

associated with clusters after the initialisation step. Then :

$$\pi_{\oplus} = \min_{C_i} \pi_{C_i}^I.$$

Example 2 Let us consider again our example. In tables 5 and 6, we provide possibility distributions associated with the two clusters after the initialization step (bold column). They are obtained as follows :

$$\pi_{C_1}^I(ABC) = \min(\pi_{\text{N}\oplus}(B | A), \pi_{\text{N}\oplus}(A | C)),$$

and

$$\pi_{C_2}^I(BCD) = \min(\pi_{\text{N}\oplus}(D | B), \pi_{\text{N}\oplus}(C | D)).$$

We already computed $\pi_{\text{N}\oplus}(a_1 b_2 c_1 d_1) = .4$. We have:

$$\pi_{C_1}^I(a_1 b_2 c_1) = .4 \text{ and } \pi_{C_2}^I(b_2 c_1 d_1) = .9. \text{ Hence:}$$

$$\pi_{\text{N}\oplus}(a_1 b_2 c_1 d_1) = \min(\pi_{C_1}^I(a_1 b_2 c_1), \pi_{C_2}^I(b_2 c_1 d_1)).$$

Table 5. Possibility distributions associated with the cluster $C_1 = \{A, B, C\}$

A	B	C	$\pi_{C_1}^I(ABC)$	$\pi_{C_1}^S(ABC)$
a_1	b_1	c_1	.8	.8
a_1	b_1	c_2	.4	.4
a_1	b_2	c_1	.4	.4
a_1	b_2	c_2	.4	.4
a_2	b_1	c_1	.6	.6
a_2	b_1	c_2	.6	.6
a_2	b_2	c_1	.7	.7
a_2	b_2	c_2	.7	.7

Table 6. Possibility distributions associated with the cluster $C_2 = \{D, B, C\}$

B	C	D	$\pi_{C_2}^I(BCD)$	$\pi_{C_2}^S(BCD)$
b_1	c_1	d_1	.9	.8
b_1	c_1	d_2	.9	.7
b_1	c_2	d_1	1	.6
b_1	c_2	d_2	.4	.4
b_2	c_1	d_1	.9	.7
b_2	c_1	d_2	.8	.7
b_2	c_2	d_1	1	.7
b_2	c_2	d_2	.4	.4

• **Step 2 (Updating of separators):** Each cluster computes its possibility distribution and sends it to the adjacent separator. The separator's distribution, denoted $\pi_{S_{ij}}^{t+1}$, is then updated as follows:

$$\pi_{S_{ij}}^{t+1} \leftarrow \max_{C_i \setminus S_{ij}} \pi_{C_i}^t. \quad (3)$$

• **Step 3 Updating of clusters:** Each cluster updates its possibility distribution, denoted $\pi_{C_j}^{t+1}$, when receiving a message from its adjacent separator as follows :

$$\pi_{C_j}^{t+1} \leftarrow \min(\pi_{C_j}^t, \pi_{S_{ij}}^t). \quad (4)$$

Steps 3 and 4 are repeated until the junction tree is globally stable (or consistent), namely adjacent clusters should have same marginal distributions with respect to common variables, namely:

$$\max_{C_i \setminus S_{ij}} \pi_{C_i}^t = \pi_{S_{ij}}^{t+1} = \max_{C_j \setminus S_{ij}} \pi_{C_j}^t$$

Proposition 6 Let $\mathbb{N}\oplus$ be the result of merging \mathbb{N}_i 's given in Proposition 4. Let $\pi_{C_i}^S$ be the possibility distribution associated with clusters after applying steps 2 and 3 until the stabilisation of the junction tree. Then :

$$\pi_{\oplus} = \min_{C_i} \pi_{C_i}^S$$

Example 3 Let us consider again our example. Tables 5 and 6 give the possibility distributions associated with clusters C_1 and C_2 after applying steps 2 and 3 until reaching stabilization. We can check that the marginal distribution on $\{B, C\}$ is the same in π_{C_1} and π_{C_2} .

We can also check that $\pi_{\mathbb{N}\oplus}(a_1 b_2 c_1 d_1) = \min(\pi_{C_1}^S(a_1 b_2 c_1), \pi_{C_2}^S(b_2 c_1 d_1)) = .4$ since we have: $\pi_{C_1}^S(a_1 b_2 c_1) = .4$ and $\pi_{C_2}^S(b_2 c_1 d_1) = .7$.

The following shows that marginals can be locally recovered:

Proposition 7 When the junction tree is consistent, computing $\Pi(A = a)$ consists in selecting any cluster containing A and marginalizing Π_{C_i} on A :

$$\Pi(A = a) = \Pi_{C_i}(A = a).$$

4.2 Dealing with subnormalized distribution

When merging multiple source information, we often meet the problem of conflicts. This is reflected by the fact that the fused possibility distribution $\pi_{\mathbb{N}\oplus}$ is subnormalized. We denote by $Cons(\pi)$ the consistency degree of a possibility distribution, defined by :

$$Cons(\pi) = \max_{\omega \in \Omega} \pi(\omega).$$

In possibility theory, the normalization procedure is given as follows:

$$\pi^{normalized}(\omega) = \begin{cases} \pi(\omega) & \text{if } \pi(\omega) < Cons(\pi) \\ 1 & \text{otherwise} \end{cases}$$

Let JT be the stable junction tree associated with fused (cyclic or not) possibilistic network. Let us denote by $\pi_{C_i}^{normalized}$ be the result of normalizing possibility distributions associated with clusters. Then we have:

Proposition 8 Let $\mathbb{N}\oplus$ be the result of merging \mathbb{N}_i 's given in Proposition 4. Then :

$$\pi_{\oplus}^{normalized} = \min_{C_i} \pi_{C_i}^{normalized}$$

The idea of the proof is to first note that for a stable junction tree, the consistency degree associated with clusters is the same, and is equal to the one associated with π_{\oplus} . Let us denote $Best(\pi)$ be the set of elements in π having their value equal to $cons(\pi)$. Then we can easily check that for

each $\omega \in \Omega$, $\omega \in Best(\pi_{\oplus})$ if and only if $\omega \in \times_{C_i} Best(\pi_{C_i})$, where $\times_{C_i} Best(\pi_{C_i})$ is the cartesian product of elements in $Best(\pi_{C_i})$. Namely, elements of $Best(\pi_{\oplus})$ are exactly those obtained by only considering elements of $Best(\pi_{C_i})$. Hence increasing the degree of elements of $Best(\pi_{C_i})$ (for all C_i 's) to 1 guarantees that only elements of $Best(\pi_{\oplus})$ will be increased to one, and the others remain unchanged. In our example, to locally renormalize π_{\oplus} , it is enough to let $\pi_{C_1}^S(a_1 b_1 c_1) = \pi_{C_2}^S(b_1 c_1 d_1) = 1$.

5 conclusion

This paper has proposed a fusion of possibilistic networks. We first showed that fusing possibilistic networks with the same structure can be easily achieved. Starting from the observation that any possibilistic networks can be equivalently extended (by adding variables or links), we suggested a method to fuse any possibilistic networks. These results also show that if the fused graph is singly connected graph, then the propagation algorithm is polynomial (wrt number of variables) even if the possibility distribution is subnormalized. They also show that the graphs associated with possibilistic networks can have cycles and hence are not restricted to be DAGS.

Acknowledgments: This work is supported by the national project ANR Blanc MICRAC (Modèles informatiques et cognitifs du raisonnement causal).

REFERENCES

- [1] C. Baral, S. Kraus, J. Minker, V.S. Subrahmanian, Combining knowledge bases consisting in first order theories. Computational Intelligence, 8(1), 45-71, 1992.
- [2] S. Benferhat, F. Titouna (2005) "Min-based possibilistic networks", Fourth Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2005), pp 553-558, 2005
- [3] S. Benferhat, D. Dubois, H. Prade (1997) From semantic to syntactic approaches to information combination in possibilistic logic. In Aggregation and Fusion of Imperfect Information, Physica Verlag, pp. 141-151.
- [4] S. Benferhat and D. Dubois and L. Garcia and H. Prade (2002). On the transformation between possibilistic logic bases and possibilistic causal networks. International Journal of Approximate Reasoning, volume 29, pp. 135-173.
- [5] D. Dubois and H. Prade (1994) Possibility theory and data fusion in poorly informed environments. Journal of Control Engineering Practice, volume 2(5), pages 811-823.
- [6] S. Konieczny, R. Pino Pérez. On the logic of merging. In Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 488-498, 1998.
- [7] F. V. Jensen, Introduction to Bayesian networks, UCL Press, University college, London, 1996.
- [8] J. Lin, Integration of weighted knowledge bases, Artificial Intelligence 83, 363-378, 1996.
- [9] C. Borgelt, J. Gebhardt, and Rudolf Kruse, Possibilistic Graphical Models, Proc. of ISSEK'98 (Udine, Italy), 1998.
- [10] D. Dubois, J. Lang and H. Prade, Possibilistic logic, In Handbook of logic in Artificial intelligence and logic programming Oxford University press, Vol. 3, 439-513, 1994.
- [11] P. Fonck, Propagating uncertainty in a directed acyclic graph, IPMU'92, 17-20, 1992.
- [12] J. Gebhardt and R. Kruse, Background and perspectives of possibilistic graphical models, Procs. of ECAI-SQARU/FAPR'97, Berlin, 1997.
- [13] E. Hisdal, Conditional possibilities independence and noninteraction, Fuzzy Sets and Systems, Vol. 1, 1978.

Compiling possibilistic knowledge bases

Salem BENFERHAT¹ and Henri PRADE²

Abstract. Possibilistic knowledge bases gather propositional formulas associated with degrees belonging to a linearly ordered scale. These degrees reflect certainty or priority, depending if the formulas encode pieces of beliefs or goals to be pursued. Possibilistic logic provides a simple format that turns to be useful for handling qualitative uncertainty, exceptions or preferences. The main result of the paper provides a way for compiling a possibilistic knowledge base in order to be able to process inference from it in polynomial time. The procedure is based on a symbolic treatment of the degrees under the form of sorted literals and on the idea of forgetting variables. The number of sorted literals that are added corresponds exactly to the number of priority levels existing in the base, and the number of binary clauses added in the compilation is also equal to this number of levels. The resulting extra compilation cost is very low.

1 Introduction

Possibilistic knowledge bases [Dubois *et al.* 1994] offer a representation framework for the qualitative handling of uncertainty or for the encoding of prioritized formulas expressing preferences. A key feature of possibilistic logic is its capability to deal with inconsistent knowledge bases. Indeed a possibilistic knowledge base is associated with a level of inconsistency, and the set of formulas whose associated level is strictly greater than this inconsistency level are safe from inconsistency, while the formulas with a level smaller or equal are ignored in the inference process. Adding new pieces of knowledge to the base may increase this inconsistency level, which provides a mechanism for encoding nonmonotonic reasoning [Benferhat *et al.* 1999]. Clearly, the computation of the inconsistency level in possibilistic logic inference, as well as for related consequence relations that are able to also take advantage of some of the ignored formulas (drawn in inconsistencies) [Benferhat *et al.* 1999], is a key issue.

Two related entailment relations can be defined from a possibilistic knowledge base. The first one yields the classical consequences of the subpart of the base made of the formulas whose certainty levels are strictly above the inconsistency level of the base, the second one yields the same consequences together with an associated certainty level (which is the greatest possible one compatible with the information in the base).

The computational complexity of possibilistic logic is the one of classical logic multiplied by the logarithm of the number of distinct levels used in the base [Lang2000]. As in classical logic,

the compilation of possibilistic knowledge bases is a key issue in order to be able to process inference in a polynomial time. The method proposed in this paper takes advantage of ideas recently introduced in [Benferhat and Prade2005] for handling an extension of possibilistic logic allowing for the symbolic handling of partial preorders between the levels. However, the case of possibilistic logic, where there is a complete preorder between the levels, requires a particular adaptation of these ideas. A key issue not considered in [Benferhat and Prade2005] is then to compute to what level a formula can be inferred. Possibilistic logic levels are encoded by means of propositional symbols. Then the possibilistic logic base we start with is turned into a new base where on the one hand the original clauses are augmented with this encoding of their levels, and where on the other hand as many binary clauses as there are certainty levels are added to the base, thus for a very small extra computational cost. This rewriting makes possible the design of an efficient original algorithm for computing the result of a possibilistic inference with its associated degree in polynomial time.

The paper is organized as follows. Section 2 provides a short background on possibilistic logic, while Section 3 introduces the symbolic encoding of levels. Section 4 presents the main results for computing of plausible inference, using DNF forms and the forgetting variable technique. Section 5 describes the general procedure for computing compiled possibilistic bases. Sections 6 and 7 show that the computation of possibilistic entailment with the level associated to the consequent can be achieved in polynomial time, and provide the algorithms for doing it.

2 Brief background on possibilistic logic

We start with a brief refresher on possibilistic logic (for more details see [Dubois *et al.* 1994]). Let V be a set of propositional variables. Let L_V be a propositional language built from V using the propositional connectors \wedge, \vee, \neg . A possibilistic logic formula is a pair made of a classical logic formula and a degree a expressing certainty or priority. The degree $a \in (0, 1]$ of a formula p is interpreted as the lower bound of a necessity measure N , i.e., the possibilistic logic expression (p, a) is understood as $N(p) \geq a$. Since $N(p \wedge q) = \min(N(p), N(q))$ it is always possible to put a possibilistic formula (p, a) under the form of a conjunction of clauses (p_i, a) where $p \equiv \wedge_i p_i$. In the following, ordinary propositions are denoted by lower case letters p, q, r, \dots , numerical degrees (belonging to the interval $[0, 1]$) are denoted by lower case letters from the beginning of the alphabet a, b, c, \dots . Anyway since the degrees play an ordinal role only, they could be replaced by the values in any linearly ordered scale.

The basic inference rule in possibilistic logic put in clausal form

¹ CRIL-CNRS, Université d'Artois, rue Jean Souvraz 62307 Lens Cedex, France. email: benferhat@cril.univ-artois.fr

² IRIT (CNRS-UPS), 118, Route de Narbonne 31062 Toulouse Cedex 04, France. email: prade@irit.fr

is the resolution rule: $(\neg p \vee q, a); (p \vee r, b) \vdash (q \vee r, \min(a, b))$. Classical resolution is retrieved when all the degrees are equal to 1.

Definition 1 Let $\Sigma = \{(p_i, a_i) : i = 1, n\}$ be a possibilistic knowledge base. The level of inconsistency of Σ is defined as:

$$\text{Inc}(\Sigma) = \max\{a : \Sigma_a \vdash \perp\}$$

(by convention $\max(\emptyset) = 0$), where :

$$\Sigma_a = \{p_i : (p_i, a_i) \in \Sigma, \text{ and } a_i \geq a\}.$$

It can be shown that $\text{Inc}(\Sigma) = 0$ iff $\Sigma^* = \{p_i : (p_i, a_i) \in \Sigma\}$ is consistent in the usual sense.

Refutation can be easily extended to possibilistic logic. Proving (p, a) from Σ amounts to adding $(\neg p, 1)$, put in clausal form, to Σ , and using the above rule repeatedly until getting $\Sigma \cup \{(\neg p, 1)\} \vdash (\perp, a)$. Clearly, we are interested here in getting the empty clause \perp with the greatest possible degree or level, i.e., $\text{Inc}(\Sigma \cup \{(\neg p, 1)\}) \geq a$. Indeed, the conclusion (p, a) is valid only if $a > \text{Inc}(\Sigma)$. More formally, the following defines the possibilistic inference:

Definition 2 Let Σ be a possibilistic knowledge base, and p be a propositional formula. Then :

- p is a possibilistic consequence of Σ , denoted by $\Sigma \models_{\pi} p$, iff $\text{Inc}(\Sigma \cup \{(\neg p, 1)\}) > \text{Inc}(\Sigma)$.
- p is a possibilistic consequence of Σ to degree a , also denoted by $\Sigma \models_{\pi} (p, a)$, iff $\text{Inc}(\Sigma \cup \{\neg p\}) > \text{Inc}(\Sigma)$ and $a = \text{Inc}(\Sigma \cup \{\neg p\})$.

Thus inferring from Σ inconsistent is equivalent to infer from the consistent subbase $\{p_i : (p_i, a_i) \in \Sigma, \text{ and } a_i > a\}$. The following illustrative example is used in the whole paper.

Example 1 Let $\Sigma = \{(\neg q \vee r, .8), (q, .6), (e, .6), (\neg r, .5), (q, .3)\}$. We have: $\text{Inc}(\Sigma) = .5$. Now, let us check if r is a possibilistic consequence of Σ and to what degree. It is enough to compute:

$$\text{Inc}(\Sigma \cup \{\neg r\}) = .6 > \text{Inc}(\Sigma).$$

Hence, r is a possibilistic consequence of Σ and its degree of inference is equal to .6.

Semantic aspects of possibilistic logic, including soundness and completeness results with respect to the above syntactic inference, are presented in [Dubois et al. 1994]. Semantically, a possibilistic knowledge base $\Sigma = \{(p_i, a_i) : i = 1, n\}$ is understood as a complete pre-ordering on the set of interpretations:

$\omega >_{\Sigma} \omega'$ iff $\max\{a_i : (p_i, a_i) \in \Sigma \text{ and } \omega' \not\models p_i\} > \max\{a_j : (p_j, a_j) \in \Sigma \text{ and } \omega' \not\models p_j\}$. Thus ω is all the less plausible as it falsifies formulas of higher degrees.

3 Symbolic encoding of possibilistic knowledge

Let $\Sigma = \{(p_i, a_i) : i = 1, \dots, n\}$ be a standard possibilistic knowledge base. Let $H = \{a_1, \dots, a_n\}$ be the set of numerical degrees or levels in Σ . Each symbol in H takes its value in the interval $[0, 1]$. We assume without loss of generality that $a_1 \geq a_2 \geq \dots \geq a_n$.

The first step in the symbolic encoding of a possibilistic knowledge base is to associate to each numerical degree a_i in the knowledge base a propositional symbol denoted by the corresponding capital letter A_i . We denote by S the set of propositional symbols associated with H (with $S \cap V = \emptyset$). Let L_S be the propositional language built from S using the propositional connectors \wedge, \vee, \neg .

We also need to encode the ordering relation between degrees. A constraint $a \geq b$ is translated into $\neg A \vee B$ in agreement with the fact that a and b are lower bounds (of a necessity measure) and thus a refers to the set of numbers $[a, 1[$ and $[a, 1] \subseteq [b, 1[$ holds iff $a \geq b$. The translation of $a \geq b$ into $\neg A \vee B$ can be read as as "if the situation is at least very abnormal (A), it is at least abnormal (B)" (indeed, the greater a , the more certain p in (p, a) , and the more exceptional a situation where p is false).

If two formulas have the same degree a , they will be associated with the same symbol A . It agrees with the equivalence $\{(p, a), (q, a)\}$ and $\{(p \wedge q, a)\}$ in possibilistic logic.

Definition 3 A total order $a_1 \geq a_2, a_2 \geq a_3, \dots, a_{n-1} \geq a_n$ between numerical degrees present in Σ is encoded by the following set of $(n-1)$ binary clauses: $\{\neg A_i \vee A_{i+1} : i = 1, \dots, n - 1\}$.

Completely certain formulas, such as tautologies, are supposed to have a level equal to $a_0 = 1$. In fact, the corresponding literal A_0 will be equivalent to \perp . Then $\forall i \geq 1, \neg A_0 \vee A_i$ holds. As suggested in the introduction, the idea is to manipulate numerical degrees as formulas [Benferhat and Prade2005]. Thus, a possibilistic formula (p, a) is associated with the classical clause $p \vee A$ where A means something as "the situation is A -abnormal". Thus $p \vee A$ means p is true or the situation is abnormal. Note that $p \vee A_0 \equiv p$.

The following definition gives the propositional logic encoding of possibilistic knowledge base:

Definition 4 Let $\Sigma = \{(p_i, a_i) : i = 1, n\}$ be a possibilistic knowledge base. Let A_i be a propositional symbol associated with a_i . Then the propositional base associated with Σ , denoted by K_{Σ} , is defined by :

$$K_{\Sigma} = \{p_i \vee A_i : (p_i, a_i) \in \Sigma \text{ and } A_i \text{ is associated with } a_i\} \\ \cup \{\neg A_i \vee A_{i+1} : i = 1, \dots, n - 1\}.$$

As can be seen from definition 4, the size of K_{Σ} is exactly the one of Σ increased by $n - 1$ binary clauses, if there are n uncertainty levels used. So the cost of this rewriting is very low.

On our running example, this rewriting gives:

Example 2 Let $\Sigma = \{(\neg q \vee r, .8), (q, .6), (e, .6), (\neg r, .5), (q, .3)\}$.

Let A, B, C, D be four propositional (representing symbolic degrees) associated respectively with the four degrees present in the possibilistic knowledge base, namely : .8, .6, .5, .3.

Using the above definition, the propositional knowledge base associated with the knowledge base Σ is :

$$K_{\Sigma} = \{A \vee \neg q \vee r, B \vee q, B \vee e, C \vee \neg r, D \vee q, \neg A \vee B, \neg B \vee C, \neg C \vee D\}.$$

The three last binary clauses are used to encode the facts that the degree associated with A is greater than B , the one of B is greater than the one of C , and the one of C is greater than the one of D .

It is easy to check that the resolution rule now reads $\neg p \vee q \vee A, p \vee r \vee B \vdash q \vee r \vee A \vee B$, and $\min(a, b)$ translates into $A \vee B$ with $A \vee B \equiv A$ if $b \geq a = \min(a, b)$ (and indeed $\neg B \vee A, A \vee B \vdash A$).

4 A refresher on forgetting variables

This section recalls the notion of forgetting variables, which are very useful for characterizing possibilistic inferences (see for instance [Lang et al. 2003, Darwiche and Marquis2004] for more details on forgetting variables). Forgetting a variable p from a set of propositional formulas K comes down to remove any reference of p in K .

Definition 5 Let p be a propositional symbol of V . Then :

$$\text{ForgetVariable}(K, p) = K_{p=\perp} \vee K_{p=\top}$$

$K_{p=\perp}$ (resp. $K_{p=\top}$) is the knowledge base obtained from K by replacing p by false (resp. true). To forget a set of variables, we forget variable by variable, namely if X denotes a set of variables, then: $\text{ForgetVariable}(K, X) = \text{ForgetVariable}(\text{ForgetVariable}(K, p), X - \{p\})$.

It is also possible to only forget literals (atoms or negated atoms):

Definition 6 Let l be a literal. Then :

$$\text{ForgetLiteral}(K, l) = K_{l=\top} \vee (\neg l \wedge K)$$

Some properties of ForgetVariable ([Darwiche and Marquis2004], [Lang et al.2003]), viewing a base as a conjunct of its formulas are:

- (1) $\text{ForgetVariable}(p \vee q, X) = \text{ForgetVariable}(p, X) \vee \text{ForgetVariable}(q, X)$.
- (2) if q does not contain any variable of X , then $\text{ForgetVariable}(p \wedge q, X) = q \wedge \text{ForgetVariable}(p, X)$

ForgetLiteral satisfies (1) and (2), which is enough for the purpose of the paper. The computation of forgetting variables can be efficiently achieved if a propositional knowledge base is in a form such as DNF (disjunctive normal form) or d-DNNF forms. Indeed, forgetting a variable in a DNF amounts to forget it in each term, and forgetting it in a term amounts just to suppress the term. This clearly shows that this is polynomial in time. A similar procedure applies as well to d-DNNF format. This format known as Deterministic, Decomposable Negation Normal Form, which has been proposed recently [Darwiche2004], is a compact format, and has allowed the computation of generally intractable logical queries in time polynomial in the form size. An algorithm has been presented in [Darwiche2004] for compiling Conjunctive Normal Forms into d-DNNF directly. Our approach can clearly take advantage of this format as well.

In the following, we denote $\text{DNF}(K)$ the result of putting K under DNF form. It is represented as a unique formula.

5 Characterising plausible inferences

This section proposes a characterisation of plausible inference using forgetting variables. We first provide a symbolic computation of the inconsistency degree of a knowledge base, then we present some propositions that characterize the compiled possibilistic knowledge base.

5.1 A symbolic computation of inconsistency degrees

This section presents the characterization of the inconsistency degree of Σ using forgetting variables on K_Σ . We first start with the case where Σ is consistent. This is stated by the following proposition:

Proposition 1 Let Σ be a possibilistic knowledge base, and K_Σ be the propositional knowledge base associated with Σ using definition 4. Let NegS be the set of negative literals in K_Σ issued from S . Let $F = \text{ForgetVariable}(K_\Sigma, V)$. Then Σ is consistent if and only if $\text{ForgetLiteral}(F, \text{NegS})$ is a tautology.

Assume that Σ is consistent. The idea of the proof is first to note that $\{A_1 \vee p_1, \dots, A_n \vee p_n\}$ is equivalent to :

$$\bigvee_{I \subseteq \{1, \dots, n\}} (\wedge_{i \in I} A_i) (\wedge_{j \in \text{complement}(I)} p_j),$$

where $\text{complement}(I)$ is the set of elements in $\{1, \dots, n\}$ that are not in I .

In particular, this equivalent form of $\{A_1 \vee p_1, \dots, A_n \vee p_n\}$ contains the term $p_1 \wedge \dots \wedge p_n$. This also means that the equivalent form of Σ contains the term $\neg A_1 \wedge \dots \wedge \neg A_{n-1} \wedge p_1 \wedge \dots \wedge p_n$, which is consistent. Hence forgetting V and negated literals from this term leads to a tautology. Hence forgetting forgetting V and negated literals from K_Σ also leads to a tautology. The converse can also be shown in a similar way.

Now, let us analyse the case where Σ is inconsistent. The following proposition provides the characterisation:

Proposition 2 Let Σ be a possibilistic knowledge base, and K_Σ be a propositional knowledge base associated with Σ using definition 4. Let NegS be the set of negative literals in K_Σ issued from S . Let $F = \text{ForgetVariable}(K_\Sigma, V)$. Then Σ is inconsistent to a degree a_i if and only if $\text{ForgetLiteral}(F, \text{NegS})$ is equivalent to $A_i \wedge \dots \wedge A_n$.

The idea of the proof is the following: if Σ is inconsistent to a degree a_i , then this means that $\{p_1, \dots, p_i\}$ is inconsistent. Hence, $\{A_1 \vee p_1, \dots, A_i \vee p_i\}$ implies $\{A_1 \vee \dots \vee A_i\}$. From $\{A_1 \vee \dots \vee A_i\}$ and $\{\neg A_1 \vee A_2, \dots, \neg A_{i-1} \vee A_i\}$ we conclude A_i . And from A_i and $\{\neg A_i \vee A_{i+1}, \dots, \neg A_{n-1} \vee A_n\}$ we conclude $A_i \wedge \dots \wedge A_n$. Therefore, all formulas of the form $A_j \vee p_j$ for $j \geq i$ can be removed from K_Σ . And K_Σ is equivalent to : $\{\neg A_1 \vee A_2, \dots, \neg A_{i-2} \vee A_{i-1}, A_1 \vee p_1, \dots, A_{i-1} \vee p_{i-1}, A_i, \dots, A_n\}$.

This knowledge base is consistent. Hence, forgetting variables of V and negated literals of S leads to $A_i \dots A_n$.

Example 3 Let us consider again our example, where we have :

$\Sigma = \{(\neg q \vee r, .8), (q, .6), (e, .6), (\neg r, .5), (q, .3)\}$, and,
 $K_\Sigma = \{A \vee \neg q \vee r, B \vee q, B \vee e, C \vee \neg r, D \vee q, \neg A \vee B, \neg B \vee C, \neg C \vee D\}$.

We already showed that Σ is inconsistent and its inconsistency degree is equal to .5. Let us show that Forgetting variables of V and negative literals of S in K_Σ is equivalent to $C \wedge D$. Remember that the symbols A, B, C, D are associated respectively with the degrees .8, .6, .5, .3.

Note that $A \vee \neg q \vee r, B \vee q$ implies $A \vee r \vee B$. Moreover, $A \vee r \vee B$ and $C \vee \neg r$ implies $A \vee B \vee C$. Lastly, $A \vee B \vee C$ and $\{\neg A \vee B, \neg B \vee C\}$ gives C . Again : C and $\neg C \vee D$ gives D .

Therefore, K_Σ is equivalent to :

$$K_\Sigma \equiv \{A \vee \neg q \vee r, B \vee q, B \vee e, \neg A \vee B, C, D\}.$$

Let us compute the DNF form of K_Σ , we get:

$$\text{DNF}(K_\Sigma) \equiv (\neg A \wedge r \wedge C \wedge D \wedge q \wedge e) \vee (A \wedge B \wedge C \wedge D) \vee (\neg q \wedge B \wedge C \wedge D) \vee (r \wedge B \wedge C \wedge D)$$

Let us now forget symbols of V but also negative literals of S .

- Forgetting $\{r\}$ gives :

$$\text{ForgetVariable}(K_\Sigma, \{r\}) \equiv (\neg A \wedge C \wedge D \wedge q \wedge e) \vee (B \wedge C \wedge D).$$

- Forgetting $\{r, q\}$ gives :

$$\text{ForgetVariable}(K_\Sigma, \{r, q\}) \equiv (\neg A \wedge C \wedge D \wedge e) \vee (B \wedge C \wedge D).$$

- Forgetting $\{r, q, e\}$ gives :

$$\text{ForgetVariable}(K_\Sigma, \{r, q, e\}) \equiv (\neg A \wedge C \wedge D) \vee (B \wedge C \wedge D).$$

- Lastly, forgetting negative literals of S gives :

$$\begin{aligned} \text{ForgetLiteral}(\text{ForgetVariable}(K_\Sigma, \{r, q, e\}), \{\neg A, \neg B, \neg C\}) \\ \equiv (C \wedge D) \vee (B \wedge C \wedge D) \equiv C \wedge D. \end{aligned}$$

Hence, we get the expected result.

5.2 Characterizing compiled possibilistic knowledge bases

This section characterises the compiled possibilistic knowledge bases. More precisely, given a possibilistic knowledge base, we are interested in characterizing a propositional knowledge base, denoted by $\text{Compiled}(\Sigma)$ such that:

$$\Sigma \models_{\pi} p \text{ iff } \text{Compiled}(\Sigma) \vdash p.$$

Of course, we require that the inference from $\text{Compiled}(\Sigma)$ should be efficient (polynomial).

Again we start with the case where Σ is consistent. This is stated by the following proposition.

Proposition 3 *Let Σ be a consistent possibilistic knowledge base, and K_{Σ} be a propositional knowledge base associated with Σ using definition 4. Then: $\text{Compiled}(K_{\Sigma}) = \text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_n \wedge \text{DNF}(K_{\Sigma}), S)$.*

The idea of the proof is quite immediate. Note first that $\text{DNF}(K_{\Sigma})$ contains a term $\neg A_1 \wedge \dots \wedge \neg A_n \wedge p_1 \wedge \dots \wedge p_n$. All other terms contain at least a positive symbol of S . Hence, adding $\neg A_1 \wedge \dots \wedge \neg A_n$ to $\text{DNF}(K_{\Sigma})$ is equivalent to the term : $\neg A_1 \wedge \dots \wedge \neg A_n \wedge p_1 \wedge \dots \wedge p_n$, since all other terms will become inconsistent.

The following proposition gives the case where Σ is inconsistent.

Proposition 4 *Let Σ be an inconsistent possibilistic knowledge base, and K_{Σ} be a propositional knowledge base associated with Σ using definition 2. Assume that Σ is consistent to a degree a_i . Then: $\text{Compiled}(K_{\Sigma}) = \text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_{i-1} \wedge \text{DNF}(K_{\Sigma}), S)$.*

Example 4 Let us consider again our example.

We recall that :

$$\begin{aligned} \text{DNF}(K_{\Sigma}) &\equiv (\neg A \wedge r \wedge C \wedge D \wedge q \wedge e) \vee (A \wedge B \wedge C \wedge D) \\ &\vee (\neg q \wedge B \wedge C \wedge D) \vee (r \wedge B \wedge C \wedge D) \end{aligned}$$

Let us compute

$$\text{Compiled}(K_{\Sigma}) \equiv \text{ForgetVariable}(\neg A \wedge \neg B \wedge \text{DNF}(K_{\Sigma}), S).$$

We have $\neg A \wedge \neg B \wedge \text{DNF}(K_{\Sigma})$ is equivalent to : $(\neg A \wedge \neg B \wedge r \wedge C \wedge D \wedge q \wedge e)$

Hence, forgetting variables of S gives :

$$\text{Compiled}(K_{\Sigma}) \equiv r \wedge q \wedge e.$$

6 Procedure for computing compiled possibilistic base

The above propositions are very important since they provide an efficient way to draw possibilistic conclusions from Σ . The procedure for checking if a proposition p can be derived from Σ can be described as follows:

The following proposition shows that the compiled knowledge base obtained at step 6 allows us to recover all possibilistic consequences of Σ .

Procedure 1: Computation of the compiled possibilistic base

Data: A possibilistic knowledge base Σ .

Result: A compiled base $\text{Compiled}(\Sigma)$

begin

Step 1: Transform Σ into K_{Σ}

Step 2: Put K_{Σ} into a DNF (or d-DNNF) form

Step 3 Forget variables of V from K_{Σ}

Step 4 Forget negated atoms of S from K_{Σ} .

Step 5 Compute inconsistency degree A_i

Step 6

if Σ is consistent then

$\text{Compiled}(\Sigma) = \text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_n \wedge \text{DNF}(K_{\Sigma}), S)$

else

/* Σ is inconsistent to a degree a_i

$\text{Compiled}(\Sigma) = \text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_{i-1} \wedge \text{DNF}(K_{\Sigma}), S)$

end

Proposition 5 *Let $\text{Compiled}(\Sigma)$ be the propositional base obtained at end of step 6 of the above algorithm. p is a possibilistic consequence of Σ iff $\text{Compiled}(\Sigma) \vdash p$.*

The proof is immediate using Propositions 1-4.

Example 5 Let us consider again our example. From previous computations we have:

$$\text{Compiled}(K_{\Sigma}) \equiv r \wedge q \wedge e$$

We can check that for all propositional formula p , p is a possibilistic consequence of Σ if and only if p is classical consequence of $\text{compiled}(K_{\Sigma})$, with the advantage that $\text{compiled}(K_{\Sigma})$ is in DNF form, hence the inference is polynomial. For instance, we have already shown (example 1) that r is a possibilistic consequence of Σ . This obviously follows from $\text{compiled}(K_{\Sigma})$.

Proposition 6 Steps 3–6 of the above algorithm are achieved in a polynomial time, with respect to the number of terms (or conjuncts) in the compiled base (which is in DNF form).

Indeed, steps 3–4 can be obtained in a linear time, since the knowledge bases are in DNF form from which forgetting operations are polynomials [Darwiche and Marquis 2004]. Step 4 which concerns the computation of inconsistency degree can also be done in a polynomial time thanks to Propositions 1 and 2. Indeed, it is enough to check if $A_1 \wedge \dots \wedge A_n$ is equivalent $\text{ForgetLiteral}(F, \text{Neg}S)$ for $i = 1, \dots, n$ with $F = \text{ForgetVariable}(K_{\Sigma}, V)$.

7 Computing weighted consequences

This section proposes an algorithm to compute the degree of inference associated with a propositional formula p , once Procedure 1 has been used for compiling Σ and establishing p as a consequence of Σ . First, we need the following result:

Proposition 7 *Let p be a propositional formula that is a consequence of $\text{Compiled}(K_{\Sigma})$. Then: $\Sigma \models_{\pi} (p_i, a_i)$ iff*

- $\text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_i \wedge A_{i+1} \dots \wedge A_n \wedge \text{DNF}(K_\Sigma), S) \vdash p_i$
- $\forall j = 1, \dots, i-1 :$
 $\text{ForgetVariable}(\neg A_1 \wedge \dots \wedge \neg A_j \wedge A_{j+1} \dots \wedge A_n \wedge \text{DNF}(K_\Sigma), S) \not\vdash p_i$

The idea of the proof is that when we add a positive literal A_i then all propositional formulas of levels $a_k \leq a_i$ are ignored. Indeed, we have A_i and $\{\neg A_i \vee A_{i+1}, \dots, \neg A_{n-1} \vee A_n\}$ implies $\{A_{i+1}, \dots, A_n\}$. Hence the set of formulas $\{A_{i+1} \vee p_{i+1}, \dots, A_n \vee p_n\}$ will be subsumed by $\{A_{i+1}, \dots, A_n\}$.

Given this proposition, the procedure for computing the degree associated with a possibilistic consequence is rather straightforward. It works iteratively, thanks to Proposition 7, since the level of p corresponds to the first A_i such that p can be proved from formulas with levels greater than a_i . So at the beginning, except those with level 1 (encoded by A_0) such as tautologies, are ignored. Then, adding $\neg A_i$ makes formulas in the corresponding stratum active in the inference process. Clearly, the procedure remains polynomial.

Procedure 2: Computing the symbolic weight associated with conclusions

Data: $F = \text{DNF}(K_\Sigma)$

A propositional formula p which is consequence of $\text{Compiled}(\Sigma)$

Result: A symbolic degree associated with p

begin

```

    k ← 0 ;
    X ← {A1, ..., An} ;
    while  $\text{ForgetVariable}(X \cup F, S) \not\vdash p$  do
        k ← k + 1;
        X ← X - {Ak} ;
        X ← X ∪ {¬Ak} ;
    return Ak
end

```

Example 6 Let us consider again our example. We already checked that r is a possibilistic consequence of Σ to a degree .6. We also checked that $\text{Compiled}(K_\Sigma) \vdash r$. Let us apply the above procedure to get the symbolic degree associated with r .

We recall that :

$$F = \text{DNF}(K_\Sigma) \equiv (\neg A \wedge r \wedge C \wedge D \wedge q \wedge e) (A \wedge B \wedge C \wedge D) \vee (\neg q \wedge B \wedge C \wedge D) \vee (r \wedge B \wedge C \wedge D)$$

- At the beginning, we have $k=0$ and $X=\{A, B, C, D\}$. Then we can check that $\text{ForgetVariable}(X \cup F, S)$ is equivalent to a tautology.
- At the next iteration, we have $k=1$ and $X=\{\neg A, B, C, D\}$. Then $X \cup F$ is equivalent to:
 $(\neg A \wedge r \wedge C \wedge B \wedge D \wedge q \wedge e)$
 $\vee (\neg A \wedge \neg q \wedge B \wedge C \wedge D) \vee (\neg A \wedge r \wedge B \wedge C \wedge D)$

Then $\text{ForgetVariable}(X \cup F, S)$ is equivalent to $a: \neg q \vee r$ from which r cannot be derived.

- At the second iteration, we have $k=1$ and $X=\{\neg A, \neg B, C, D\}$. Then $X \cup F$ is equivalent to:
 $(\neg A \wedge \neg B \wedge r \wedge C \wedge D \wedge q \wedge e)$
 $\vee (\neg A \wedge \neg B \wedge \neg q \wedge C \wedge D) \vee (\neg A \wedge \neg B \wedge r \wedge C \wedge D)$
- Then $\text{ForgetVariable}(X \cup F, S)$ is equivalent to $q \wedge r$ from which r can be derived. Hence, the symbolic degree associated with r is B , which is indeed associated with the degree 0.6.

Proposition 8 The above algorithm, which computes the degree associated with a possibilistic conclusion, is achieved in a polynomial time, with respect to the number of terms (or conjuncts) in the compiled base (which is in DNF form).

8 Conclusion

The paper has proposed an approach for handling possibilistic inference (including the computation of the formula levels) in polynomial time thanks to a compilation procedure that takes advantage of a propositional treatment of formula levels, and to the use of a forgetting variable procedure. Note also that the proposed compilation procedure cannot be derived from the one used by [Darwiche and Marquis2004] for penalty logic, which relies on summations and is quantitative in nature, while possibilistic logic is qualitative and uses max and min operations. Moreover, the results are also relevant for processing other inconsistency-tolerant inferences that extend possibilistic inference and are also based on the computation of inconsistency levels [Benferhat *et al.* 1999].

A line for further research is the handling of hypothetical reasoning in this framework. Indeed, by moving literals in the level slots, namely the possibilistic formula $(\neg p \vee q, a)$ is equivalent to $(q, \min(P, a))$ where $P = 1$ or 0 depending if p is assumed true or false, one can develop a possibilistic counterpart of ATMS [de Kleer1986]. It would be interesting to combine the ideas presented here, the handling of more general symbolic levels (as in [Benferhat and Prade2005]), and consequence-finding algorithms (e.g. [Inoue1991]) in that perspective.

Acknowledgments: This work is supported by the national project ANR Blanc MICRAC (Modèles informatiques et cognitifs du raisonnement causal).

REFERENCES

- [Benferhat and Prade2005] S. Benferhat and H. Prade. Encoding formulas with partially constrained weights in possibilistic-like many-sorted propositional logic. In *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 1281–1286, 2005.
- [Benferhat *et al.* 1999] Salem Benferhat, Didier Dubois, and Henri Prade. An overview of inconsistency-tolerant inferences in prioritized knowledge bases. In *Fuzzy Sets, Logic and Reasoning about Knowledge*. D. Dubois, H. Prade, E.P. Klement (Eds.), pages 395–417, 1999.
- [Darwiche and Marquis2004] A. Darwiche and P. Marquis. Compiling propositional weighted bases. *Artif. Intell.*, 157(1-2):81–113, 2004.
- [Darwiche2004] A. Darwiche. New advances in compiling cnf into decomposable negation normal form. In *Procs. of Europ. Conf. on Art. Intell. (ECAI2004)*, pages 328–332, 2004.
- [de Kleer1986] Johan de Kleer. An assumption-based TMS and extending the ATMS. *Artificial Intelligence*, 28(2):127–162 and 163–196, 1986.
- [Dubois *et al.* 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.
- [Inoue1991] K. Inoue. onsequence-finding based on ordered linear resolution. In *12th Inter. JOint Conf. on artificial Intelligence (IJCAI'91)*, Sydney, pages 158–164, 1991.
- [Lang *et al.* 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence - formula-variable independence and forgetting. *J. of Art. Intell. Research*, 18:391–443, 2003.
- [Lang2000] Jérôme Lang. Possibilistic logic: complexity and algorithms. In D. M. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5: Algorithms for Uncertainty and Defeasible Reasoning, pages 179–220. Kluwer Academic, Dordrecht, 2000.

Improving Bound Propagation

Bozhena Biduk¹ and Rina Dechter²

Abstract. This paper extends previously proposed bound propagation algorithm [11] for computing lower and upper bounds on posterior marginals in Bayesian networks. We improve the bound propagation scheme by taking advantage of the directionality in Bayesian networks and applying the notion of relevant subnetwork. We also propose an approximation scheme for the linear optimization subproblems. We demonstrate empirically that while the resulting bounds loose some precision, we achieve 10-100 times speedup compared to original bound propagation using a simplex solver.

1 Introduction

Using Bayesian networks to model the problems arising in practical applications requires answering queries regarding the probability of an outcome given observations, namely, computing posterior marginals. Computing exact posteriors is NP-hard. Computing bounds on posterior marginals is a special case of approximating posterior marginals with the desired degree of precision which is also NP-hard [6, 2]. Previously proposed methods include bounded conditioning [7], search with conflict-counting [14], "context-specific" bounds [15], *large deviation bounds* for layered networks [8, 9], bounds for goal directed queries [12], and a scheme bounding exact computation in bucket elimination [10]. None of the methods dominates the rest as they offer different accuracy and speed trade-offs.

We focus on a recently proposed **bound propagation** (*BdP*) algorithm [11], applicable to both Bayesian networks and Markov random fields. The algorithm iteratively solves a linear optimization problem for each variable such that the minimum and maximum of the objective function correspond to lower and upper bounds on the variable's posterior marginals. The lower and upper bounds are initialized to 0 and 1 respectively. When algorithm solves minimization/maximization LP problem, the lower and upper bounds are updated. The bounds are updated repeatedly until they converge. The performance of the scheme was demonstrated in [11] on the example of Alarm network, Ising grid, and regular bi-partite graphs.

The performance of bound propagation is tied to the network structure, namely, the Markov blanket of each variable. Its computation time increases exponentially with Markov blanket size. Hence, it is well suited for problems with regular network structure, having large induced width but bounded Markov blanket size (e.g., grids).

In our work here, we improve the performance of *BdP* by exploiting the global network properties, namely, restricting the Markov blanket of a node to its relevant subnetwork, resulting in substantial gains in accuracy and speed. Further, since bound propagation yields linear optimization subproblems that fall into a category of fractional packing and covering problems, known to be hard for simplex

method, we propose a fast approximate algorithm for solving the LP problems without using simplex method. Although many schemes have been developed for approximately solving linear programming problems [5], they usually solve packing-only or covering-only problems and do not include the additional constraints present in bound propagation. Hence, we propose our own solution obtained by relaxing the original problem until it can be solved exactly using a greedy algorithm. We investigate empirically the trade-offs in bounds interval length and time.

2 Background

2.1 Belief Networks

We use upper case letters without subscripts, such as \bar{X} , to denote sets of variables and an upper case letter with a subscript, such as X_i , to denote a single variable. We use a lower case letter with a subscript, such as x_i , to denote an instantiated variable. $\mathcal{D}(X_i)$ denotes the domain of the variable X_i . We will use \bar{x} to denote an instantiation of a set of variables $\bar{x} = \{x_1, \dots, x_i, \dots\}$ and $\bar{x}_{-i} = \bar{x} \setminus x_i$ to denote \bar{x} with element x_i removed.

Definition 1 (belief networks) Let $\bar{X} = \{X_1, \dots, X_n\}$ be a set of random variables over multi-valued domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_n)$. A belief network (BN) is a pair (G, P) where G is a directed acyclic graph on \bar{X} and $P = \{P(X_i|pa_i)\}$ is the set of conditional probability tables (CPTs) associated with each X_i . The parents of a variable X_i together with its children and parents of its children form a Markov blanket ma_i of node X_i . A network is **singly-connected** (also called a **poly-tree**), if its underlying undirected graph has no cycles. The queries over singly-connected network can be processed in time linear in the size of the network [13].

2.2 Bound Propagation

Bound propagation (*BdP*) [11] is an iterative algorithm that utilizes the local network structure to formulate a linear optimization problem. For each variable $X_i \in \bar{X}$ the minimum and maximum of the objective function correspond to the upper and lower bounds on posterior marginals $P(x_i|e)$. Let $\bar{Y} = \{Y_1, \dots, Y_k\}$ denote the Markov blanket of node X_i . The idea is to compute posterior marginals using decomposition:

$$P(x_i|e) = \sum_{\bar{y}} P(x_i|\bar{y}, e)P(\bar{y}|e) \quad (1)$$

Given its Markov blanket, the probability distribution of X_i is independent from the rest of the variables in the network so that $P(x_i|\bar{y}, e) = P(x_i|\bar{y})$. Hence, we can rewrite Eq. (1) as:

$$P(x_i|e) = \sum_{\bar{y}} P(x_i|\bar{y})P(\bar{y}|e) \quad (2)$$

¹ Donald Bren School of Information and Computer Science, University Of California Irvine, CA 92697-3425, USA, email: bbidyuk@ics.uci.edu

² email: dechter@ics.uci.edu

Here, $P(x_i|\bar{y})$ is an entry in the probability table of X_i conditioned on the instantiation of variables in its Markov blanket which can be computed as follows [13]:

$$P(x_i|pa_i, \cup_j(ch_j \cup pa_j)) = \alpha P(x_i|pa_i) \prod_j P(ch_j|pa_j)$$

where α is a normalization constant. The probabilities $P(\bar{y}|e)$ are unknown, but we know that the sum of all probabilities equals 1:

$$\sum_{y_1, \dots, y_k} P(y_1, \dots, y_k|e) = 1 \quad (3)$$

Further, $\forall Y_j \in \bar{Y}, \forall y_j \in \mathcal{D}(Y_j), \sum_{\bar{y}_j, Y_j=y_j} P(\bar{y}|e) = P(y_j|e)$. Denoting arbitrary lower and upper bounds on $P(y_j|e)$ by $P^L(y_j|e)$ and $P^U(y_j|e)$ respectively, we can write:

$$P^L(y_j|e) \leq \sum_{y \setminus y_j, Y_j=y_j} P(y_1, \dots, y_k|e) \leq P^U(y_j|e) \quad (4)$$

Hence, for each variable X_i , we have a linear optimization problem with the objective function $P(x_i|e)$ defined in Eq. (2) and constraints defined in Eq. (3) (sum-to-1 constraint) and Eq. (4). For each $\bar{y} \in \mathcal{D}(\bar{Y})$, the $P(x_i|\bar{y})$ is an objective function coefficient and $P(\bar{y}|e)$ is a variable. The number of variables is exponential in the size of the Markov blanket. The number of constraints equals $1 + \sum_j |\mathcal{D}(Y_j)|$ since there are $|\mathcal{D}(Y_j)|$ constraints for each Y_j .

Example 1 Let $ma_i = \{A, B\}$ where $\mathcal{D}(A) = \{0, 1\}$ and $\mathcal{D}(B) = \{0, 1, 2\}$. Let $P(x_i|A, B)$ be defined as follows: $P(x_i|0, 0) = 0.1$, $P(x_i|0, 1) = 0.2$, $P(x_i|0, 2) = 0.3$, $P(x_i|1, 0) = 0.4$, $P(x_i|1, 1) = 0.5$, and $P(x_i|1, 2) = 0.6$. Then, denoting $P_{pq} = P(x_i|p, q)$, the objective function of the LP problem for x_i can be defined as follows:

$$f = 0.1P_{00} + 0.2P_{01} + 0.3P_{02} + 0.4P_{10} + 0.5P_{11} + 0.6P_{12}$$

$$\text{s.t. } P_{00} + P_{01} + P_{02} + P_{10} + P_{11} + P_{12} = 1$$

$$\begin{aligned} P^L(a=0|e) &\leq P_{00} + P_{01} + P_{02} \leq P^U(a=0|e) \\ P^L(a=1|e) &\leq P_{10} + P_{11} + P_{12} \leq P^U(a=1|e) \\ P^L(b=0|e) &\leq P_{00} + P_{10} \leq P^U(b=0|e) \\ P^L(b=1|e) &\leq P_{01} + P_{11} \leq P^U(b=1|e) \\ P^L(b=2|e) &\leq P_{02} + P_{12} \leq P^U(b=2|e) \end{aligned}$$

Initializing all bounds $P^L(X_i|e)$ and $P^U(X_i|e)$ to 0 and 1, the algorithm solves the linear minimization and maximization problems for each value $x_i \in \mathcal{D}(X_i)$ of each variable $X_i \in \bar{X}$ and updates the bounds. With every iteration, the bounds get closer to the posterior marginals or do not change. The process is iterated until convergence. The variable processing order does not affect the results although it may affect the number of iterations needed to converge.

Since the number of variables in the LP problems grows exponentially with the size of the Markov blanket, algorithm *BdP* is feasible only for networks having bounded Markov blanket size e.g. Ising grid an regular two-layer networks explored in [11]. Applied to Alarm network without evidence, *BdP* obtained small bounds interval for several nodes but could not obtain good bounds for root nodes 11, 12, 13, 14, although their relative subnetworks are singly-connected and, hence, the posteriors equal the priors. The latter shows the weakness of *BdP* in that it may not compute tight bounds even in a singly-connected network.

3 Improving *BdP* Performance

In this section we describe how to improve the performance of *BdP* by exploiting global network structure and how to obtain quick bounds using a simple greedy algorithm instead of a simplex solver.

3.1 Exploiting Network Structure

The performance of bound propagation can be improved also by identifying the irrelevant child nodes and restricting the Markov blanket of X_i to its relevant subnetwork.

Definition 2 (Relevant Subnetwork) An irrelevant (barren) node of a node X_i is a child node Y_j that is not observed and does not have observed descendants. The relevant subnetwork of X_i is a subnetwork obtained by removing all irrelevant nodes in the network.

Removing irrelevant nodes (and their parents) from Markov blanket whenever possible yields a smaller effective Markov blanket and, thus, a smaller LP problem with fewer variables. Also, if the relevant subnetwork of node X_i is singly-connected then its posteriors should be computed exactly and fixed.

We denote as *BdP+* the bound propagation algorithm that takes advantage of the network structure as described above. Although proposed improvements are straight forward, the gains in accuracy and speed are significant in practice, as we show empirically.

3.2 Managing Resources

In order to limit *BdP* demands for memory and time, we can bound the maximum length of the Markov conditional probability table by a constant k and, thus, the maximum number of variables in a linear optimization problem. For variables, whose Markov blanket size exceeds the maximum, their lower and upper bound values remain equal to their input values (usually, 0 and 1). The resulting algorithm *BdP(k)* is then parametrized by k .

Since the bounds of variable X_i are used to define constraints of the neighboring variables, fixing the bounds of X_i to their input values will result in a more relaxed LP formulation. Thus, the bounds of neighboring nodes are likely to be less tight as well, affecting, in turn, the bounds of their neighbors. Hence, the effect of fixing bounds of X_i can propagate throughout the network resulting in the overall larger average bounds interval. As k increases, the computation time will increase, but the bounds will become tighter.

3.3 Approximating the LP in Bound Propagation

In this section, we propose an algorithm for solving the linear optimization problem approximately, instead of using a simplex solver.

In large Bayesian networks, we may need to solve linear optimization problems thousands of times. Using the simplex method then becomes impractical time-wise. In general, the linear optimize problems which are formulated in bound propagation fall into a class of linear packing and covering problems which are known to be especially challenging for the simplex method [5]. The standard fractional packing and covering problem can be defined as follows:

$$\min c^T \bar{x} \quad (5)$$

$$A\bar{x} \geq l \quad (6)$$

$$B\bar{x} \leq m \quad (7)$$

$$x \geq 0 \quad (8)$$

Without Eq. (7), it is a **fractional covering** problem. Without Eq. (6), it is a **fractional packing** problem. The BdP (and $BdP+$) linear optimization problems have both packing and covering components with the special properties that $A=B$ and A is a zero-one matrix. Still, the problem remains hard. Existing approximate algorithms solve either packing or covering problem, but not both [5]. The LP formulation in BdP is complicated further by having an additional sum-to-1 constraint. Hence, we resort to solving a relaxed problem.

We considered two relaxations of the LP formulation in BdP . First, we relaxed the problem to an instance of a fractional knapsack packing which can be solved exactly in a greedy fashion [16]. In this case, we maintain the sum-to-1 constraint, but drop the lower bound constraints completely and replace the upper bounds on sums of variables with the derived upper bounds on individual variables. Namely, for each variable $P(\bar{y}|e)$ participating in $|Y|$ constraints, we set:

$$P(\bar{y}|e) \leq UB_{\bar{y}} = \min_j P^U(y_j|e) \quad (9)$$

We obtain an optimal solution to the fractional knapsack packing by first ordering the variables by their coefficient (from maximum to minimum for maximization and from minimum to maximum for minimization) and then assigning each variables its maximum value until the sum of all values equals 1. The complexity of the algorithm is $O(n \log n)$, where n is the number of variables, due to the complexity of sorting.

The second relaxation is more constrained. We maintain the sum-to-1 constraint and the lower and upper bound constraint pertaining to one variable in the Markov blanket of X_i . We drop the remaining lower bounds and use remaining upper bounds to set upper bounds on individual variables. Consider the example presented previously with a Markov blanket consisting of two nodes A and B . Maintaining the constraints associated with variable A , the resulting relaxed optimization problem can be expressed as:

$$f = 0.1P_{00} + 0.2P_{01} + 0.3P_{02} + 0.4P_{10} + 0.5P_{11} + 0.5P_{12}$$

$$\text{s.t. } P_{00} + P_{01} + P_{02} + P_{10} + P_{11} + P_{12} = 1$$

$$\begin{aligned} P^L(a=0|e) &\leq P_{00} + P_{01} + P_{02} \leq P^U(a=0|e) \\ P^L(a=1|e) &\leq P_{10} + P_{11} + P_{12} \leq P^U(a=1|e) \\ 0 &\leq P_{00}, P_{10} \leq P^U(b=0|e) \\ 0 &\leq P_{01}, P_{11} \leq P^U(b=1|e) \\ 0 &\leq P_{02}, P_{12} \leq P^U(b=2|e) \end{aligned}$$

The domains of the constraints w.r.t. just one Markov variables Y_j are disjoint. Hence, the problem can be treated as an instance of the fractional packing with multiple knapsacks, each having a separate set of packing materials and an individual capacity bound. If it was not for the sum-to-1 constraint, we could solve each knapsack packing problem independently. Nevertheless, we can show that the problem can be solved optimally by a greedy algorithm.

We describe the idea of the algorithm on the example of maximization problem. Similar to fractional packing with 1 knapsack, we first order nodes by their objective function coefficient value from the largest to smallest. We initialize all node values to 0. Then, we make two passes through the list. The first time, we satisfy all lower bound requirements. Namely, we increment each node value until either its upper bound is reached or the lower bound $L(y_j)$ of the equation in which it participates is satisfied. During a second pass, we increment each variable value until either the variables' upper bound or the upper bound $U(y_j)$ of the equation in which it participates is reached or

the sum of all variables equals 1. Since we cannot predict which variable $Y_j \in Y$ will yield the LP relaxation with the smallest maximum of the objective function, we repeat computation for each $Y_j \in Y$ and pick the smallest maximum of the objective function.

The solution to the minimization problem is the same except nodes are ordered by their objective function coefficient value from smallest to largest. We prove formally that the algorithm finds an optimal solution in [4]. The total complexity is $O(|Y| \cdot n \log n)$, $n = |\mathcal{D}(Y)|$. We call the bound propagation scheme with an approximate LP algorithm as $ABdP+$.

4 Experiments

We compare empirically the performance of the original bound propagation algorithm BdP , modified $BdP+$ that restricts the Markov blanket of a node to its relevant subnetwork, and a modified bound propagation scheme using the approximate algorithm for solving linear programming subproblems, namely, $ABdP+$.

4.1 Methodology

We measure the quality of the bounds via the average length of the interval between lower and upper bound:

$$\bar{I} = \frac{\sum_i \sum_{D(x_i)} (P^U(x_i|e) - P^L(x_i|e))}{\sum_i |D(x_i)|} \quad (10)$$

We compute approximate posterior marginal as the midpoint between lower and upper bound in order to show whether the bounds are well-centered around the posterior marginal $P(x|e)$. Namely:

$$\hat{P}(x|e) = \frac{P^U(x|e) + P^L(x|e)}{2} \quad (11)$$

and then measure average absolute error Δ with respect to $\hat{P}(x|e)$:

$$\Delta = \frac{\sum_i \sum_{D(x_i)} |P(x_i|e) - \hat{P}(x_i|e)|}{\sum_i |D(x_i)|} \quad (12)$$

We control the time and memory of bound propagation by restricting the maximum length k of a conditional probability table over the Markov blanket of a variable. The maximum Markov CPT length tested was $k=2^{19}$ (the size of the CPT with 19 bi-valued variables) when the computation demands exceeded available memory.

We report all results for BdP , $BdP+$, and $ABdP+$ schemes "upon convergence" or after 20 iterations, whichever occurs first, so that the computation time is a function of k and number of iterations needed to converge. We implemented bounds propagation algorithm using simplex solver from COIN-OR libraries [1]. The experiments were conducted on 1.8Ghz CPU with 512 MB RAM.

Our benchmarks are 8 networks from Bayesian Network Repository (<http://www.cs.huji.ac.il/labs/compbio/> Repository/) and 3 networks for pedigree analysis, gen44, gen50, and gen51 (from <http://bioinfo.cs.technion.ac.il/superlink/ExpF.html>). Benchmarks' properties are specified in Table 4.1. In gen44, gen50, and gen51 evidence is pre-defined. In all other benchmarks, the results are averaged over 20 instances of each network instantiated with different evidence. In most networks the evidence variables are selected at random among leaf nodes. In cpcs422b, the evidence is selected at random among all variables.

Table 1. Benchmarks' characteristics: N -number of nodes, w^* -induced width, T_{BE} -exact computation time via bucket elimination.

network	N	w^*	T_{BE}
Alarm	37	4	0.01 sec
cpcs54	54	15	1 sec
cpcs179	179	8	2 sec
cpcs360b	360	21	20 min
cpcs422b	422	22	50 min
gen44	873	N/A	47 min
gen50	547	N/A	>6 hrs
gen51	1218	N/A	> 7 hrs

4.2 Results

4.2.1 BdP vs. BdP+

In Tables 2 and 3 we report the average error, average bounds interval, and computation times for BdP and $BdP+$ as a function of $k=2^m$ for $14 \leq m \leq 19$. Each row corresponds to a set of experiments with a single benchmark with a fixed k . Note that, as k increases, the computation time of both BdP and $BdP+$ increases fast, while the bounds interval decreases only a little.

Table 2. Average error Δ , length of the bounds interval \bar{I} , and computation time for $BdP(k)$ and $BdP+(k)$ in networks without evidence.

	k	BdP(k)			BdP+(k)		
		\bar{I}	Δ	time	\bar{I}	Δ	time
Alarm	16384	0.6369	0.1677	14	0.0753	0.0076	0.1
cpcs54	16384	0.4247	0.0229	24	0.0907	0.0049	0.1
	32768	0.4173	0.0224	72	0.0907	0.0049	0.1
	262145	0.4154	0.0221	265	0.0907	0.0049	0.1
cpcs179	16384	0.5759	0.2213	30	0.0006	0.00002	0.3
	32768	0.5759	0.2213	30	0.0006	0.00002	0.3
cpcs360b	16384	0.1505	0.0649	64	0.0006	0.0002	1.2
	32768	0.1485	0.0641	80	0.0006	0.0002	1.2
cpcs422b	16384	0.2339	0.0756	79	0.0082	0.0008	8
	32768	0.2329	0.0751	88	0.0082	0.0008	8

Table 3. Average error Δ , length of the bounds interval \bar{I} , and computation time for $BdP(k)$ and $BdP+(k)$ in networks with evidence.

	k	BdP(k)			BdP+(k)		
		\bar{I}	Δ	time	\bar{I}	Δ	time
Alarm $ E =3-6$	16384	0.8276	0.2661	13	0.6376	0.2084	5.3
	65536	0.8276	0.2661	13	0.6376	0.2084	5.3
cpcs54 $ E =2-6$	16384	0.6021	0.0448	46	0.2638	0.0138	6.6
	32768	0.5986	0.0445	64	0.2637	0.0138	7.4
	65536	0.5957	0.0440	98	0.2637	0.0138	10
cpcs179 $ E =12-24$	16384	0.6034	0.2227	30	0.1525	0.0456	20
	32768	0.6034	0.2227	30	0.1502	0.0450	24
	65536	0.5983	0.2214	90	0.1237	0.0365	130
cpcs360b $ E =11-23$	16384	0.3375	0.1423	68	0.0637	0.0247	15
	32768	0.3370	0.1419	85	0.0554	0.0215	24
	65536	0.1430	0.3367	120	0.0500	0.0192	36
	131072	0.1430	0.3366	128	0.0429	0.0160	80
cpcs422b $ E =6-11$	16384	0.3373	0.1200	34	0.2140	0.0665	24
	32768	0.3287	0.1081	80	0.2034	0.0617	74
	65536	0.3171	0.1023	317	0.1815	0.0522	256

$BdP+$ always computes tighter bounds and requires less computation time than BdP . The performance gap is wider in the networks without evidence (Table 2), where the Markov blanket of each node,

restricted to its relevant subnetwork, contains node's parents only and $BdP+$ converges after one iteration when processing nodes in topological order. For the largest benchmark, cpcs422b, with 422 nodes and $w^*=21$, the average bounds interval length is 0.23 for BdP and 0.008 for $BdP+$. At the same time, BdP computations take 190 sec while $BdP+$ only takes 16 sec.

In benchmarks with evidence, reported in Table 3, $BdP+$ computation time increases but its bounds remain superior to BdP . Consider the results for cpcs360b network with 360 nodes and $|E|$ ranging from 11 to 23. For $k=16384$, $BdP+$ computes the average bounds interval of length 0.06 within 15 seconds. BdP average bounds interval equals 0.34 and requires 68 seconds. We observed similar results for other benchmarks.

4.2.2 Approximating LP

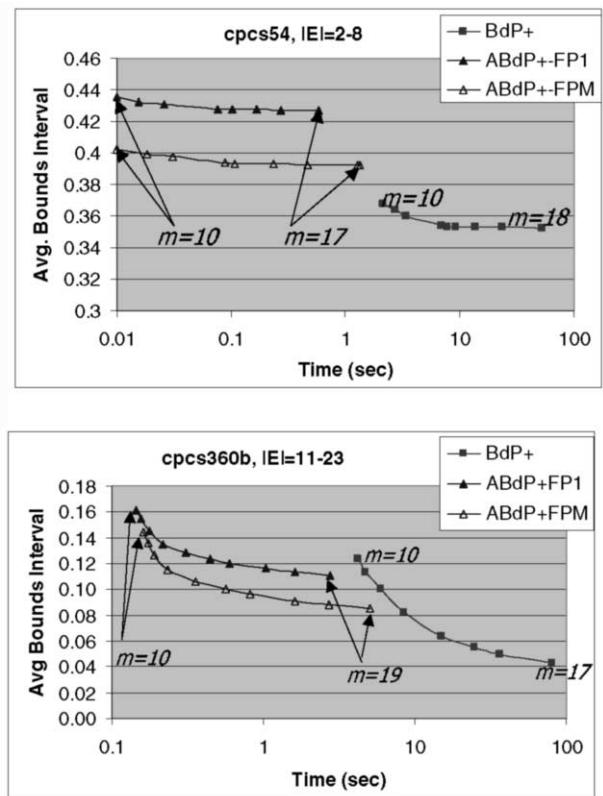


Figure 1. Bounds interval for $BdP+$ and $ABdP+$ optimizing LP by fractional packing with 1 ($ABdP+FP1$) and many ($ABdP+FPM$) knapsacks. Each data point corresponds to a value of input parameter m that bounds the maximum Markov table length $k=2^m$.

We compare the performance of $BdP+$ and $ABdP+$ with two different approximation schemes, $FP1$ and FPM . $FP1$ solves the fractional packing with 1 knapsack; FPM solves the fractional packing and covering problem with multiple knapsacks.

The results shown in Figure 1 for cpcs54 and cpcs360b are typical. The average bounds interval is shown as a function of time which, in turn, depends on the input parameter $k=2^m$. Hence, each point on the graph corresponds to a particular value of m . First, we observe that $ABdP+$ with FPM (denoted $ABdP+FPM$) substantially outperforms $ABdP+$ with $FP1$ (denoted $ABdP+FP1$) in

both benchmarks. *FPM* incurs negligible amount of computation overhead but computes considerably tighter bounds. *BdP+* outperforms both *ABdP+* with enough time, but *ABdP+* is more efficient at the start. For the same k , *BdP+* requires an order of magnitude more time than *ABdP+*. In cpcs54, for $k=2^{10}$, *ABdP+FPM* computes bounds in <0.01 seconds while *BdP+* requires a few seconds. We observe similar result in case of cpcs360b. Roughly, *BdP+* requires as much time to compute bounds for $k=2^{10}$ as *ABdP+FPM* for $k=2^{17}-2^{19}$. As a result, *BdP+* begins to outperform *ABdP+FPM* only after ≈ 2 seconds in cpcs54 and only after 10 seconds in cpcs360b. The overall improvement in *BdP+* bounds over *ABdP+FPM* in cpcs54 is minor; the difference between their average bounds intervals is only about 0.04 which is $\approx 1\%$ of the interval length. In cpcs360b, the *BdP+* bounds interval after 100 seconds is a factor of 2 smaller, but the computation time is an order of magnitude greater.

Table 4. The # of probabilities in each pedigree network (gen44, gen50, gen51) with bounds interval \bar{I} in different ranges, *BdP+* vs. *ABdP+FPM*.

	gen44		gen50		gen51	
	BdP+	ABdP+	BdP+	ABdP+	BdP+	ABdP+
time	54	0.8	66	8.4	140	0.7
$0.5 \leq \bar{I} < 1$	1312	1324	1114	1114	1428	1432
$0.4 \leq \bar{I} < 0.5$	14	6	6	6	36	36
$0.3 \leq \bar{I} < 0.4$	6	2	8	10	2	0
$0.2 \leq \bar{I} < 0.3$	2	2	6	4	4	2
$0.1 \leq \bar{I} < 0.2$	10	10	6	6	6	6
$0.01 \leq \bar{I} < 0.1$	0	0	10	10	12	12
$0.001 \leq \bar{I} < 0.01$	0	0	2	2	5	5
$0 \leq \bar{I} < 0.001$	14	14	18	18	174	174

In pedigree networks, both schemes performed poorly computing $\bar{I} < 0.5$ only for 5-15% of posteriors using maximum $k=2^{19}$. Yet, importantly, *ABdP+* obtained similar results to *BdP+* in a fraction of time. Table 4 specifies the number of posterior marginals with bounds in different ranges obtained by both algorithms. In gen44, both *BdP+* and *ABdP+FPM* computed $\bar{I} \in [0, 0.001]$ for 14 values and $\bar{I} \in [0.1, 0.2]$ for another 10 values. However, *BdP+* requires 54 sec whereas *ABdP+FPM* takes 0.8 sec. In gen50, the results only differ in that 2 values with *BdP+* bounds interval in range $[0.2, 0.3]$ have *ABdP+FPM* bounds interval in range $[0.3, 0.4]$, but *BdP+* takes 66 sec while *ABdP+FPM* computes bounds in 8.4 sec. Similarly, in gen51, algorithm *ABdP+FPM* computes only 4 values less with the bounds interval in range $[0.2, 0.4]$, but completes computation in 0.7 sec while *BdP+* requires 140 sec.

5 Conclusions and Future Work

In this paper, we proposed two improvements to the bound propagation algorithm [11]. First improvement exploits the directionality of Bayesian networks, restricting the Markov blanket of a variable to its relevant subnetwork. Although the idea is straight forward, it results in substantial improvement in the accuracy of bounds while the computation time is reduced which we demonstrated empirically.

Second improvement defines an approximation algorithm for the linear optimization problems in bound propagation scheme. Namely, we proposed a relaxation of the LP problem and a greedy algorithm that can solve it exactly. Although the bounds obtained by approximate LP solver are less accurate, we reduce computation time by an order of magnitude or more. Hence, bound propagation with approximate LP solver is more practical for real-time on-line applications

where speed is of critical importance while a small loss in accuracy may be acceptable. An efficient practical strategy may be also to select a threshold value k^* and use the simplex method to solve small LP problems (Markov blanket size $< k^*$) and use an approximate method to solve large LP problems.

We used the proposed bound propagation algorithm with approximate LP solver as a plug-in in the any-time framework for computing bounds on posterior marginals [3], where using *BdP+* with simplex solver was not feasible time-wise. The loss of bounds accuracy was not significant since the framework focuses on enumerating high-probability cutset-tuples and only uses *ABdP+FPM* to bound the remaining probability mass of $P(e)$. We showed in [3] that any-time framework with *ABdP+FPM* outperformed *BdP+* after exploring a few hundred to a few thousand cutset tuples.

Performance of bound propagation depends on the efficiency of the linear optimization algorithm. We have looked at only two approximation schemes. Other approximation algorithms, offering different time/accuracy trade-offs, need to be investigated.

6 Acknowledgments

This work has been partially supported by the NSF grant IIS-0412854.

REFERENCES

- [1] COmputational INfrastructure for Operations Research, <http://www.coin-or.org>.
- [2] A. M. Abdelbar and S. M. Hedetniemi, ‘Approximating maps for belief networks is NP-hard and other theorems’, *Artificial Intelligence*, **102**, 21–38, (1998).
- [3] B. Bidyuk and R. Dechter, ‘An anytime scheme for bounding posterior beliefs’, in *In Proc. of 21st National Conf. on AI (AAAI)*, (2006).
- [4] B. Bidyuk and R. Dechter, ‘Improving bound propagation’, Technical report, UCI, <http://www.ics.uci.edu/~bbidyuk/bp.html>, (2006).
- [5] D. Bienstock, *Potential function methods for approximately solving linear programming problems: theory and practice*, Kluwer Academic Publishers, 2002.
- [6] P. Dagum and M. Luby, ‘Approximating probabilistic inference in Bayesian belief networks is NP-hard’, *Artificial Intelligence*, **60**(1), 141–153, (1993).
- [7] E.J. Horvitz, H.J. Suermondt, and G.F. Cooper, ‘Bounded conditioning: Flexible inference for decisions under scarce resources’, in *Workshop on Uncertainty in Artificial Intelligence*, pp. 181–193, (1989).
- [8] M. Kearns and L. Saul, ‘Large deviation methods for approximate probabilistic inference, with rates of convergence’, in *Proc. of Uncertainty in AI*, pp. 311–319. Morgan Kaufmann, (1998).
- [9] M. Kearns and L. Saul, ‘Inference in multilayer networks via large deviation bounds’, *Advances in Neural Information Processing Systems*, **11**, 260–266, (1999).
- [10] D. Larkin, ‘Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries’, in *Proc. of UAI*, pp. 346–353, (2003).
- [11] M. A. R. Leisink and H. J. Kappen, ‘Bound propagation’, *Journal of Artificial Intelligence Research*, **19**, 139–154, (2003).
- [12] M. V. Mannino and V. S. Mookerjee, ‘Probability bounds for goal directed queries in Bayesian networks’, *IEEE Transactions on Knowledge and Data Engineering*, **14**(5), 1196–1200, (September/October 2002).
- [13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.
- [14] D. Poole, ‘Probabilistic conflicts in a search algorithm for estimating posterior probabilities in Bayesian networks’, *Artificial Intelligence*, **88**(1–2), 69–100, (1996).
- [15] David Poole, ‘Context-specific approximation in probabilistic inference’, in *Proc. of Uncertainty in AI*, pp. 447–454, (1998).
- [16] R. L. Rivest T. H. Cormen, C. E. Leiserson, *Introduction to Algorithms*, MIT Press, 1990.

Logic Programs with Multiple Chances

Francesco Buccafurri and Gianluca Caminiti and Domenico Rosaci¹

Abstract. In human-like reasoning it often happens that different conditions, partially alternative and hierarchically structured, are mentally grouped in order to derive some conclusion. The hierarchical nature of such knowledge concerns with the possible failure of a chance of deriving a conclusion and the necessity, instead of blocking the reasoning process, of activating a subordinate chance. Traditional logic programming (we refer here to Answer Set Programming) does not allow us to express such situations in a synthetic fashion, since different chances of deriving a conclusion must be distributed over different rules, and conditions enabling the switching among chances must be explicitly represented. We present a new language, relying on Answer Set Programming, which incorporates a new modality able to naturally express the above features. The merits of the proposal about the capability of representing knowledge are shown both by examples and by comparisons with other existing formalisms. A translation to plain ASP is finally provided in order to give a practical tool for computing our programs, since a number of optimized ASP evaluation systems are nowadays available.

1 INTRODUCTION

In human-like reasoning it often happens that different conditions are mentally grouped in order to derive some conclusion. Such conditions represent different chances, partially overlapped, exploitable as alternative ways for proceeding in the reasoning process. Moreover, following different kinds of ordering, like reliability, simplicity, cost, and so on, different chances are hierarchically structured, and the failure of a chance of deriving a conclusion, caused by uncertainty conditions, enables the application of a subordinate chance. This way, the reasoning process is not necessarily blocked by uncertainty. Consider for example the diagnostic medical reasoning. The doctor tries to recognize a classical clinical picture, in order to derive a diagnosis, but often such a picture is unclear and incomplete. In this case he typically adopts subordinate ways for reaching the same conclusion, like consulting, ex-adiuvantibus treatments, laboratory tests and so on. Traditional logic programming (we refer here to Answer Set Programming [8]) does not allow us to express such situations in a synthetic fashion, since different chances of deriving a conclusion must be distributed over different rules, and conditions enabling the switching among chances must be explicitly represented.

In this paper, we define a language extending Answer Set Programming (ASP) by including a new modality, allowing us to represent, in a compact and natural fashion, the multi-chances form of reasoning described above. Besides ASP, our programs, called *MC-programs*, are compared also with *Inheritance Logic Programming* [5] and *Nested Logic Programming* [12], that appear to us as the best candidates for representing multiple chances. The former is chosen

among languages allowing default reasoning with exceptions and prioritized rules [2, 15, 6, 4, 9], the latter is a powerful logic language allowing us to define rules with nested if-then-else constructs. A translation to plain ASP is finally provided in order to give a practical tool for computing our programs, since a number of optimized ASP solvers [10, 14] and optimization techniques [3] are available.

In order to give the flavor of our proposal, we illustrate the following simple MC-program, describing the diagnosis of appendicitis performed by a doctor:²

$app \leftarrow (\underline{fever}, pl, pf, \underline{nausea})[wbc_test]$

The structure of the above rule, which we call MC-rule, is the following. First, it embeds a standard logic rule, i.e., $app \leftarrow \underline{fever}, pl, pf, \underline{nausea}$, whose meaning is the usual one: If the conjunction $\underline{fever}, pl, pf, \underline{nausea}$ holds, where pl (resp. pf) means pain localization (resp. pain form), then the diagnosis app (appendicitis) is derived. The MC-rule contains also a subordinate condition, namely wbc_test (denoting the result of a laboratory test counting white blood cells), whose truth is required, in order to derive app , whenever the truth assignment of the elements of the conjunction $\underline{fever}, pl, pf, \underline{nausea}$ is recognized as *uncertain*. Such a machinery models the reasoning approach of the doctor, who adopts a subordinate strategy (i.e, the laboratory test) in order to derive the diagnosis, in case the clinical picture allows him neither to exclude such a diagnosis nor to conclude it (i.e., in case of an uncertain clinical picture).

More generally, through an MC-rule of the form $head \leftarrow body[sub_cond]$, our purpose is to represent a logic rule like $head \leftarrow body$ which is not blocked every time $body$ fails (like in standard ASP), but enables the evaluation of the subordinate condition sub_cond only in case the value of $body$ reflects an uncertain situation. In our example, following a standard approach, we could certainly recognize as uncertain truth assignments just those where at least one literal among $\underline{fever}, pl, pf, \underline{nausea}$ is undefined and the others are true³. However, the notion of uncertainty we are using now is strictly related to the intended meaning of the rule. As a consequence, it might happen that other truth assignments are recognizable as uncertain, and thus, the occurrence of such assignments has to enable the evaluation of the subordinate condition wbc_test . In particular, truth assignments including some false value, might be good candidates for activating the evaluation of the subordinate condition, whenever such false values actually contain uncertain knowledge (for example, *false negatives* of a test result). Consider for example the literal *fever*. The doctor knows that in case of absence of fever (i.e.,

¹ DIMET, Università Mediterranea di Reggio Calabria, Italy, email: {bucca, gianluca.caminiti, domenico.rosaci}@unirc.it

² The same example is encoded in Section 4 both in Answer Set Programming and in Nested Logic Programming.

³ Recall that, under ASP semantics, it may happen that an intended model contains neither a given literal nor its complementary literal. Throughout this paper the perspective we use in order to capture the above issue is allowing for literals three truth values: true, false and undefined.

whenever *fever* is false) he cannot exclude the diagnosis of appendicitis (if the rest of the clinical picture holds). The same happens to the symptom *nausea*. Conversely, whenever the localization of the pain is not compatible with the diagnosis of appendicitis (i.e., *pl* is false) the diagnosis can be excluded (the same happens to the pain form). As a consequence, we should include into the set of truth assignments reflecting uncertain situations (corresponding in our example to an uncertain clinical picture), also those in which *fever* and *nausea* are false. This is done, in our formalism, by underlining such literals in the body of the rule. Observe that in case of absence of underlined literals, a conjunction of literals is recognized as uncertain exactly when it is undefined according to the standard meaning (i.e., at least one literal in the conjunction is undefined and the others are true). For instance, according to the above statements, whenever the patient has both a normal temperature, a clear localization of the pain compatible with the diagnosis of appendicitis, an unclear form of pain and, further, he does not have nausea, then the doctor recognizes an uncertain situation not allowing him to exclude the diagnosis. However, such an unclear clinical condition together with the positive result of the laboratory test *wbc_test* (which is the subordinate condition activated in case of uncertainty), allows the doctor to conclude the diagnosis. The language provides the programmer with another important tool. Some uncertainty configurations, among all possible ones, could be considered non-admissible on the basis of possible relationships among the elementary conditions of a conjunction. In our example, even though the conditions *pl* and *pf* may appear separately in admissible uncertainty configurations, the doctor thinks that they cannot be missing simultaneously. In other words, he considers very improbable that neither the localization nor the form of the pain appear in a definite way. This is modeled in our language by defining some *admissibility constraints*, associated to the conjunction of literals. In our example we have just one admissibility constraint $\{pl, pf\}$ encoding what we have described above. The set of admissibility constraints is inserted just after the conjunction on which they operate. The resulting MC-program is the following:

$$app \leftarrow (\underline{fever}, pl, pf, \underline{nausea}) \{ \{pl, pf\} \} [wbc_test] \quad (1)$$

The plan of the paper is the following. Section 2 describes syntax and semantics of the language. In Section 3 the features of the language are shown by real-life examples. In Section 4 our capability of representing knowledge is compared with other existing approaches. Section 5 describes the translation of our language into Answer Set Programming. Finally, we draw our conclusions in Section 6.

2 SYNTAX AND SEMANTICS

In this section, after a brief recall of basic concepts about *Answer Set Programming* (ASP) [8], we introduce the syntax of logic programs with multiple chances, said *MC-programs*, and then we give its semantics, that is an adaptation of the notion of *answer sets*⁴.

An *atom* is an expression $p(t_1, \dots, t_n)$, where p is a *predicate* of arity n and t_1, \dots, t_n are constants. A *literal* is either a *positive literal* a or a *negative literal* $\neg a$, where a is an atom and \neg is the *classical negation symbol*. Given a literal a , $\neg a$ is defined as $\neg p$, if $a = p$ and p if $a = \neg p$. A set L of literals is *consistent* if $\forall l \in L, \neg l \notin L$. Given a literal a , the formula *not a* is the *negation as failure* (NAF) of a ⁵.

⁴ For the sake of presentation, we refer to variable-free (also said *ground*) programs. The extension to the general case is straightforward.

⁵ Observe that the NAF of negative literals is allowed.

Definition 1 Given $m \geq 0$ and $k \geq 0$, an *MC-formula* β is a formula $(\alpha_1[\gamma_1], \dots, \alpha_m[\gamma_m])\{T_1, \dots, T_k\}$, where (1) α_j ($1 \leq j \leq m$) is (possibly the NAF of) a literal and may appear underlined (2) T_i ($1 \leq i \leq k$) is a subset of $\{\alpha_j \mid (\alpha_j \text{ is either underlined, or is not the NAF of a literal}) \wedge 1 \leq j \leq m\}$, and (3) γ_j ($1 \leq j \leq m$) is a (possibly empty) conjunction of MC-formulas. Each set T_i ($1 \leq i \leq k$) is called *admissibility constraint* and the (possibly empty) set $\{T_1, \dots, T_k\}$ of admissibility constraints of β is denoted by $S(\beta)$. γ_j ($1 \leq j \leq m$) is said the *second-chance* of α_j .

The recursive definition above is a generalization of the standard definition of the conjunction of (possibly the NAF of) literals occurring in the body of an ASP logic rule, where both (1) a second-chance is (possibly) associated to each literal, (2) literals may appear underlined and (3) a set of admissibility constraints is associated to the conjunction. As we will explain in the following, the second-chance may be viewed like a substitution of the α_j to which is applied. Such a substitution is executed only when α_j is *uncertain* (the notion of uncertainty will be introduced in the following – the underlining is relating with this notion). Due to the recursive structure of the above definition, a second-chance is, in turn, a conjunction of MC-formulas, and thus may contain further subordinate chances (i.e., the nesting of chances is allowed). Intuitively, second-chances are subordinate conditions that are required, in order to satisfy the MC-formula, whenever the basic conditions α_i s are uncertain. Actually, not all the possible configurations of values making α_i s uncertain are allowed, in order to activate the substitution of the second-chance, but only those satisfying the admissibility constraints.

Given a conjunction of MC-formulas Δ , in favor of simplicity, we often write $(\alpha_1[\gamma_1], \dots, \alpha_m[\gamma_m])\{T_1, \dots, T_k\}[\Delta]$ to denote $(\alpha_1[\gamma_1[\Delta]], \dots, \alpha_m[\gamma_m[\Delta]])\{T_1, \dots, T_k\}$. In particular, if $\gamma_1 = \dots = \gamma_m = \Delta$, we write $(\alpha_1, \dots, \alpha_m)\{T_1, \dots, T_k\}[\Delta]$ to denote $(\alpha_1[\Delta], \dots, \alpha_m[\Delta])\{T_1, \dots, T_k\}$. Clearly, square brackets of empty second-chances are omitted. Examples of MC-formulas are $\beta_1 = (a, \text{not } b)[c]$ (denoted also by $(a[c], \text{not } b[c])$), where $S(\beta_1) = \emptyset$ and $\beta_2 = (a, b)\{a, b\}[c]$, (denoted also by $(a[c], b[c])\{a, b\}$), where $S(\beta_2) = \{a, b\}$.

Definition 2 An *MC-rule* r is a formula $a \leftarrow \beta_1, \dots, \beta_n$ ($n \geq 0$), where a is a positive literal and β_i is an MC-formula, for each $1 \leq i \leq n$. The set $\{a\}$, denoted by *head(r)*, is called the *head* of r , and the set $\{\beta_1, \dots, \beta_n\}$, denoted by *body(r)*, is called the *body* of r . An *MC-program* is a finite set of MC-rules.

Note that since an MC-formula is a generalization of a standard conjunction of (the NAF of) literals, a standard ASP rule is a special case of an MC-rule r . Informally, an MC-rule is a logic rule allowing in the body the substitution of some literal with its second-chance (under uncertainty conditions). An example of MC-rule is the expression (1) reported in Section 1. Another example of MC-rule is: $a \leftarrow b[c[d]], f[\text{not } g]$.

We introduce now the intended models of our semantics. First we need some preliminary definitions. Given an (MC-)program P , Lit^P is the set of literals occurring in P . An *interpretation* I of an (MC-)program P is a consistent subset of Lit^P . A literal a is *true* w.r.t. I if $a \in I$, it is *false* w.r.t. I if $\neg a \in I$. A literal a is *undefined* w.r.t. I if it is neither true nor false w.r.t. I . Given a literal a , a formula *not a* is *true* w.r.t. I if $a \notin I$, it is *false* w.r.t. I otherwise (observe that the NAF of a literal cannot be undefined w.r.t. a given interpretation). From now on in this section, consider given an (MC-)program P and an interpretation I of P .

We introduce now a basic notion of our framework, that is the notion of *uncertainty* of an element of an MC-formula. Indeed the mechanism of substitution of an element of an MC-formula with its second-chance is founded on this property (this will be explained in detail in the following).

Given an MC-formula $\beta = (\alpha_1[\gamma_1], \dots, \alpha_m[\gamma_m])\{T_1, \dots, T_k\}$, we say that α_j ($1 \leq j \leq m$) is *uncertain* w.r.t. the interpretation I if either (i) $\alpha_j = a \mid a \in \text{Lit}^P \wedge \{a, \neg a\} \cap I = \emptyset$, (ii) $\alpha_j = \underline{a} \mid a \in \text{Lit}^P \wedge a \notin I$, or (iii) $\alpha_j = \underline{\text{not } a} \mid a \in \text{Lit}^P \wedge a \in I$.

The definition of uncertainty introduced above arises from the following reasoning. We expect that an element α_j is uncertain if it is undefined. This is captured by the item (i) of the definition. Note that, correctly, this item does not include the case of the NAF of a literal, since such a formula cannot be undefined. Now, when we want to interpret as uncertain also the false value of α_j , we require that α_j is underlined (items (ii) and (iii)). Thanks to this mechanism, also the NAF of a literal can be uncertain, whenever it appears in an underlined element $\alpha_j = \underline{\text{not } a}$, and a is true.

Consider again the MC-formula β introduced above. $S(\beta)$ is *true* w.r.t. I if for each $T_i \in S(\beta)$ ($1 \leq i \leq k$), there exists $\alpha \in T_i$ such that α is not uncertain w.r.t. I . In words, an admissibility constraint T_i states that literals occurring in it cannot appear simultaneously in uncertainty configurations. Now we define the intended models of our semantics. First, we introduce a transformation for MC-programs which produces an ASP program.

Definition 3 We define the *MC-transformation* of the program P w.r.t the interpretation I as the ASP program \bar{P}^I , obtained from P by executing Algorithm 1.

Algorithm 1 (The MC-transformation algorithm)

```

repeat
  for each MC-rule  $r \in P$  do
    if  $\exists \beta \in \text{body}(r) \mid S(\beta)$  is false w.r.t.  $I$  then delete  $r$ 
    else for each  $\beta = (\alpha_1[\gamma_1], \dots, \alpha_m[\gamma_m])\{T_1, \dots, T_k\} \in \text{body}(r)$  do
      delete  $S(\beta)$ 
      for each  $1 \leq j \leq m$  do
        if  $\alpha_j$  is uncertain w.r.t.  $I$  then remove from  $\alpha_j$  the underlining (if any)
        if  $\gamma_j$  is not empty then replace  $\alpha_j[\gamma_j]$  by  $\gamma_j$ 
        end if
      else remove from  $\alpha_j$  the underlining (if any); replace  $\alpha_j[\gamma_j]$  by  $\alpha_j$ 
      end if
    end for
  end for
  end if
end for
until  $\forall r \in P, \forall \beta \in \text{body}(r) \exists a \in \text{Lit}^P \mid \beta = a \vee \beta = \text{not } a$ 
```

The MC-transformation of P consists of both (i) deleting all MC-rules of P whose body includes some MC-formula with false admissibility constraint, (ii) deleting all admissibility constraints from the remaining MC-formulas, (iii) for each uncertain element α_j in the body of every MC-rule, removing the underlining (if any), and replacing α_j by its second-chance (if any); (iv) for each remaining element α_j , removing the underlining (if any) and discarding its second-chance. The loop ends when only (possibly the NAF of) literals occur in P .

Definition 4 An interpretation J of the program P is an *answer set* of P if J is an answer set⁶ of \bar{P}^J .

For instance, consider the following MC-program P :

⁶ The definition of *answer sets* of an ASP program can be found in [8]. We do not report it here for space limitations. Note that, according to the definition of interpretation, we want to limit our focus only on consistent answer sets.

$$\begin{array}{ll} r_1 : a \leftarrow (b, c, e)\{\{c, e\}, \{\underline{b}, c\}\}[\text{not } d] & r_3 : \neg b \leftarrow a \\ r_2 : d \leftarrow \text{not } a & r_5 : e \leftarrow \\ r_4 : c \leftarrow \text{not } d & \end{array}$$

The intended answer sets of P are: $\{a, \neg b, c, e\}, \{d, e\}$. Indeed, given $I_1 = \{a, \neg b, c, e\}$, the MC-transformation of P w.r.t. I_1 is $\bar{P}^{I_1} = \{r'_1, r_2, r_3, r_4, r_5\}$, where r'_1 is $a \leftarrow \text{not } d, c, e$. Observe that I_1 is an answer set of \bar{P}^{I_1} thought as an ASP program. Likewise, it is easy to see that $I_2 = \{d, e\}$ is an answer set of $\bar{P}^{I_2} = \{r_2, r_3, r_4, r_5\}$ (r_1 is deleted due to the constraint $\{\underline{b}, c\}$).

Remark. Our formalism can be viewed as an extension of the ASP negation by failure. Indeed, the rule $h \leftarrow \text{not } b$ can be rewritten in our setting as $h \leftarrow \neg b[\text{true}]$. According to our semantics, h is derived if either b is false (i.e., $\neg b$ is true) or b is undefined (since, in this case the subordinate condition is activated and it corresponds to the constant *true*), exactly as the ASP rule $h \leftarrow \text{not } b$.

3 KNOWLEDGE REPRESENTATION BY MC-PROGRAMS

In this section, we show how our language can be used for naturally representing real-life situations. We show two examples, where we do not make explicit the answer sets of the programs, because they depend on the value of a number of literals, which we assume as “external variables”. However, the purpose of this section is to make evident the merits of the multi-chance constructs. Therefore in the following description we just stress the aspect of the meaning of MC-rules, instead of considering the issues typically occurring in the context of answer sets (like their multiplicity), which our language completely preserves.

A Medical Example. We consider here a fragment of a diagnostic reasoning process in a medical setting. We model, by means of the following MC-program, the reasoning steps leading to both the diagnosis of *ischemic cardiopathy* (rules $r_1 - r_4$) and the consequent therapy (rules $r_5 - r_8$).

$$\begin{array}{lll} r_1 : & \text{clin_pict} \leftarrow & \text{complete_set_of_symptoms} \\ r_2 : & \neg \text{clin_pict} \leftarrow & \text{absent_symptoms} \\ r_3 : & \text{ECG_test} \leftarrow & \text{ST_segment_anomaly} \\ r_4 : & \text{isch_cardio} \leftarrow & (\text{clin_pict}, \text{N_test}, \underline{\text{ECG_test}}[\text{PPI_test}]) \\ & & \{\{\text{clin_pict}, \text{N_test}\}\}[\text{enz_test}[\text{echo}]] \\ r_5 : & \text{active_ulcer} \leftarrow & \text{recent_symptoms}, \text{pos_case_history} \\ r_6 : & \neg \text{active_ulcer} \leftarrow & \neg \text{pos_case_history} \\ r_7 : & \text{risk_patient} \leftarrow & \text{smoker}, \text{diabetic}, \text{hypertensive} \\ r_8 : & \text{aspirin} \leftarrow & \text{isch_cardio}, \\ & & \neg \text{active_ulcer}[\text{risk_patient}] \end{array}$$

In particular, rules r_1 and r_2 specify that the patient’s clinical picture (*clin_pict*) is either (i) true, if there exists a complete set of symptoms, or (ii) false, if all symptoms are absent. Otherwise *clin_pict* is considered uncertain. Rule r_3 describes that the electrocardiogram (ECG) test (*ECG_test*) is positive if it reveals a typical anomaly of its ST-segment. Thus, by means of rule r_4 , the doctor concludes the diagnosis of ischemic cardiopathy (*isch_cardio*) in case of both a complete clinical picture, the patient’s positive reaction to a nitrate treatment (*N_test*), and a positive result of the ECG. In this case, rule r_4 behaves as the following standard ASP rule: $\text{isch_cardio} \leftarrow \text{clin_pict}, \text{N_test}, \text{ECG_test}$.

However, a possible uncertainty situation might arise from a negative ECG: The doctor knows that there are cases of ischemic cardiopathy not altering the ST-segment of the ECG (i.e. *ECG_test* is false). In such cases he applies a second-chance approach, by substituting the information given by the ECG test with that given by

another ex-adiuvantibus treatment, i.e. the administration of a protonic pump inhibitor (*PPI*_test). The patient's positive reaction to such a test, reveals a different kind of pathology. Thus, *PPI*_test is assumed true whenever symptoms persists after the administration of the inhibitor. After such a substitution, rule r_4 behaves as: $\text{isch_cardio} \leftarrow \text{clin_pict}, N_test, PPI_test$.

Unfortunately, there are other situations that may appear still unclear: (i) The clinical picture might be not complete (*clin_pict* may be undefined)⁷. (ii) The nitrate treatment might give either an unclear result (i.e., symptoms may weakly persist) or false negatives (*N_test* may be either undefined or false, respectively). (iii) Symptoms might weakly persist after the protonic-pump-inhibitor-based treatment (i.e., *PPI*_test may be undefined).

Observe that the admissibility constraint $\{\{\text{clin_pict}, N_test\}\}$ in rule r_4 means that the simultaneous occurrence of both case (i) and (ii) is not compatible with the diagnosis of ischemic cardiopathy.

In either case (i), (ii) or (iii), the doctor decides to perform a laboratory test, which is aimed to detect specific enzymes in the blood. The result of such a test is represented by *enz*_test. Accordingly, rule r_4 behaves as: $\text{isch_cardio} \leftarrow \text{enz_test}$.

However, since this test might give false negatives, the doctor evaluates the nested second-chance (*echo*), corresponding to the result of a echocardiogram. Thus, rule r_4 behaves as: $\text{isch_cardio} \leftarrow \text{echo}$.

In order to apply the therapy, the doctor reasons about the patient's history. In particular, he is interested in the possible presence of an active ulcer (rules r_5 and r_6). Rule r_5 states that the patient is affected by an active ulcer (*active_ulcer*), if both the doctor recognizes the occurrence of recent symptoms and the patient's history clearly reveals previous cases of ulcer. Rule r_6 excludes the occurrence of an active ulcer in case of a clearly negative patient's history. However, it may happen that the history reported by the patient is unclear, so that the information about such a pathology remains undefined. Moreover, rule r_7 specifies under which conditions the patient has to be considered a patient with a high cardiovascular risk (*risk_patient*).

Finally, rule r_8 describes the therapy (*aspirin*) to be administered in case of both ischemic cardiopathy and in absence of an active ulcer ($\neg\text{active_ulcer}$). However, if the latter information is undefined, then the doctor replaces the condition $\neg\text{active_ulcer}$ by *risk_patient*, i.e. he decides to administer the aspirin to the patient only if both the diagnosis is certain and the patient has a high cardiovascular risk. Accordingly, rule r_8 behaves as: $\text{aspirin} \leftarrow \text{isch_cardio}, \text{risk_patient}$. Observe that, in this case, the doctor judges the risk of a both non-adequate and non-immediate therapy for the ischemic cardiopathy to be more relevant than the probability of side effects due to the interaction of aspirin and a (possible, but not evident) active ulcer.

The Driving Licence. A guy must go to a far town for renewing his driving license. The renewal takes a few minutes, but in order to apply for it, two documents *A* and *B* are required by the administration office. Moreover, *A* and *B* are issued by two different offices, say O_A and O_B , that are placed in that town. He is told that *A* and *B* will be available in about 15 days. Meanwhile, he can check by phone for both *A* and *B* to be ready. However, it is possible that either both O_A and O_B phone lines are busy, or the officers cannot answer. The guy wants to go to the administration office only if he knows that both offices O_A and O_B have issued the respective documents. However, if after 20 days he is sure about the availability of at least one document, and he was not informed of the contrary about the other document, then he goes. This example may be represented

by the following MC-program:

```
go ← (rdy_A, rdy_B){{rdy_A, rdy_B}}[waiting(20)]
```

where we assume that *rdy_A* (resp. *rdy_B*) is derived as a fact if the guy has been told by phone that *A* (resp. *B*) is ready, $\neg\text{rdy}_A$ (resp. $\neg\text{rdy}_B$) is derived as a fact if the guy has been told by phone that *A* (resp. *B*) is not ready and *waiting(20)* is derived as a fact if the guy has waited for 20 days.

4 COMPARISON WITH OTHER APPROACHES

In this section first we compare by examples our language with plain ASP, in order to show that the introduction of the new modality in ASP gives real benefits. Then, we compare our programs also with *Inheritance Logic Programming* [5] and *Nested Logic Programming* [12], that appear, to our knowledge, as the best candidates among logic programming languages for representing multiple chances.

Plain ASP. Consider now the MC-program (1) presented at the end of Section 1 and encode the same real-life situation using Answer Set Programming. Since we need to make explicit all the possible combinations of uncertain elements of the conjunction (which number grows exponentially with the number of uncertain cases), the resulting ASP program is the following:

```
app ← fever, pl, pf, nausea
app ← not fever, pl, pf, nausea, wbc_test
app ← not fever, pl, pf, not nausea, wbc_test
app ← not fever, pl, not pf, not \pf, nausea, wbc_test
app ← not fever, not pl, not \pl, pf, nausea, wbc_test
app ← not fever, not pl, not \pl, pf, not nausea, wbc_test
app ← not fever, pl, not pf, not \pf, not nausea, wbc_test
app ← fever, pl, pf, not nausea, wbc_test
app ← fever, pl, not pf, not \pf, nausea, wbc_test
app ← fever, not pl, not \pl, pf, nausea, wbc_test
app ← fever, not pl, not \pl, pf, not nausea, wbc_test
app ← fever, pl, not pf, not \pf, not nausea, wbc_test
```

This example shows that even though our formalism does not extend the expressive power of ASP (and does not introduce asymptotic computational cost), it has evident merits as far as its capability of representing multi-chance reasoning is concerned. Concerning the complexity and the expressiveness of logic programming, see [7].

Nested Logic Programming. *Nested Logic Programming* [12] (NLP) is a class of logic programs, where arbitrarily nested expressions – formed from literals by using negation as failure, conjunction and disjunction (\wedge) – are allowed in both the bodies and heads of rules. Given an MC-program P , a suitable translation P' into the non-disjunctive fragment of NLP (with no NAF in heads of rules) exists, but we show that P' is extremely tedious to write and difficult to read. For instance, if we translate the MC-program (1) of Section 1, we obtain the following NLP program:

```
app ← (fever; not fever, wbc_test),
      (pl; not pl, not \pl, (pf; \pf), wbc_test),
      (\pf; not pf, not \pf, (pl; \pl), wbc_test),
      (nausea; not nausea, wbc_test)
```

Disjunctive Logic Programs with Inheritance. *Disjunctive Logic Programs with Inheritance* [5] ($\text{DLP}^<$) is a knowledge representation language extending disjunctive logic programming (with strong negation) by inheritance. The addition of inheritance allows us a natural representation of default reasoning with exceptions. Thus, one could argue that $\text{DLP}^<$ may be used to represent default conditions being evaluated whenever previously required conditions are uncertain. We have compared by examples our language with $\text{DLP}^<$. For space limitations we do not report details of such a comparison here.

⁷ Observe that if $\neg\text{clin_pict}$ holds, then such a case is not considered as a font of uncertainty. That is, if no symptom is present, then the doctor is able to exclude the diagnosis of ischemic cardiopathy.

Synthetically, we have tested that DLP $^<$, compared to our language, is not suitable to represent multi-chance reasoning. Indeed base conditions must be thought as exceptions and (nested) chances as defaults and, further, programs are time-wasting to write and difficult to read, since all the combinations, i.e. those resulting from the possible uncertainty of the literals enclosed in an MC-formula, must be individually considered.

5 TRANSLATION TO ASP

In this section we show that for any MC-program P , an ASP program $\Gamma(P)$ exists such that there is a one-to-one correspondence between the answer sets of P and the answer sets of $\Gamma(P)$. Such a translation allows us to evaluate any MC-program by exploiting one of the existing answer set solvers (DLV [10], Smodels [14], etc).

Given an MC-program P , we define $\Gamma(P)$ as the ASP program obtained from P by executing Algorithm 2.

Algorithm 2 (The translation algorithm)

```
for each MC-rule  $r \in P$  do
  for each MC-formula  $\beta \in body(r) \mid \beta$  is not (the NAF of) a literal do
    RESOLVE_FORMULA( $r, \beta, \bar{\beta}$ )
  end for
end for
```

The core of Algorithm 2 is the recursive procedure described by Algorithm 3, where MC-formulas are of the form $(\alpha_1[\gamma_1], \dots, \alpha_m[\gamma_m])\{T_1, \dots, T_k\}$ ($m \geq 0, k \geq 0$).

Algorithm 3 (The procedure RESOLVE_FORMULA)

```
procedure RESOLVE_FORMULA( $r, \beta, \bar{\beta}$ )
  body( $r$ ) = body( $r$ ) \ { $\beta$ } \ { $\bar{\beta}$ }
   $P = P \cup \{\bar{\beta} \leftarrow \rho_1, \dots, \rho_m, \text{not } \tau_1, \dots, \text{not } \tau_k\} \cup$ 
     $\cup \{\tau_i \leftarrow \sigma_1, \dots, \sigma_{|T_i|} \mid 1 \leq i \leq k\}$ 
  for  $1 \leq j \leq k$  do let  $c_1, \dots, c_{|T_j|}$  be a permutation of  $T_j$ 
    for  $1 \leq l \leq |T_j|$  do
      if  $(c_l = b) \wedge (b \in Lit^P)$  then  $P = P \cup \{\sigma_l \leftarrow \text{not } b, \text{not } \neg b\}$ 
      else if  $(c_l = \underline{b}) \wedge (b \in Lit^P)$  then  $P = P \cup \{\sigma_l \leftarrow \text{not } b\}$ 
      else if  $(c_l = \underline{\text{not } b}) \wedge (b \in Lit^P)$  then  $P = P \cup \{\sigma_l \leftarrow b\}$ 
      end if
    end for
  end for
  for  $1 \leq j \leq m$  do
    if  $(\alpha_j = b \vee \alpha_j = \underline{b}) \wedge (b \in Lit^P)$  then  $P = P \cup \{\rho_j \leftarrow b\}$ 
    if  $\gamma_j$  is not empty then
      if  $(\alpha_j = b) \wedge (b \in Lit^P)$  then let  $t : \rho_j \leftarrow \text{not } b, \text{not } \neg b, \gamma_j$ 
      else if  $(\alpha_j = \underline{b}) \wedge (b \in Lit^P)$  then let  $t : \rho_j \leftarrow \text{not } b, \gamma_j$ 
      end if
       $P = P \cup \{t\}; \text{ RESOLVE\_FORMULA}(t, \gamma_j, \bar{\gamma}_j)$ 
    end if
    else if  $(\alpha_j = \text{not } b \vee \alpha_j = \underline{\text{not } b}) \wedge b \in Lit^P$  then
       $P = P \cup \{\rho_j \leftarrow \text{not } b\}$ 
      if  $(\gamma_j \text{ is not empty}) \wedge (\alpha_j = \underline{\text{not } b}) \wedge (b \in Lit^P)$  then
        let  $t : \rho_j \leftarrow b, \gamma_j; \ P = P \cup \{t\}; \text{ RESOLVE\_FORMULA}(t, \gamma_j, \bar{\gamma}_j)$ 
      end if
    end if
  end for
end procedure
```

Note that for each call of the procedure, $\rho_i, \tau_i, \sigma_i, \bar{\gamma}_i(\forall i)$ and $\bar{\beta}$ are fresh literals (not already included in Lit^P). Observe that the Algorithm 2 is exponential in the maximum number, say c , of nested chances. However, c is typically bound (informally, the number of nested chances is small). Therefore it becomes meaningful the complexity analysis w.r.t. n , that is the number of literals occurring in the program, keeping constant c . It is easy to see that such a complexity is $O(n)$. Moreover, in such a case, the computational complexity still remains linear w.r.t. both the number of program rules and the maximum number of elements in the body of an MC-rule.

In the next theorem we state the equivalence between the original MC-program P and its translation $\Gamma(P)$. The proof of the theorem is omitted for space limitations.

Theorem 1 Given an MC-program P and an interpretation I of P , then I is an answer set of P iff $I \in AS^{dep}(\Gamma(P))$, where $AS^{dep}(\Gamma(P))$ is the set of answer sets of $\Gamma(P)$ where all working literals $\bar{\beta}, \rho_i, \tau_i, \sigma_i, \bar{\gamma}_i(\forall i)$ are discarded.

6 CONCLUSIONS

The paper presents a new language relying on Answer Set Programming and including some new constructs useful for naturally representing some forms of reasoning where multiple chances of deriving a given conclusion occur. Examples described in the previous sections as well as comparisons with other languages show that the above goal is satisfactorily reached. The general KR point of view adopted in this work could represent a starting point for designing more specific solutions, once a particular research setting is chosen (for example, planning [13, 16, 11, 1]). We feel this is a very interesting direction for our future research.

ACKNOWLEDGMENTS

We are grateful to Dr. Umberto Buccafurri for the useful suggestions given us during the preparation of medical examples.

REFERENCES

- [1] M. Balduccini, M. Gelfond, R. Watson, and M. Nogueira, ‘The USA-Advisor: A Case Study in Answer Set Planning.’, in *LPNMR*, volume 2173 of *LCNS*, pp. 439–442. Springer, (2001).
- [2] G. Brewka and T. Eiter, ‘Preferred Answer Sets for Extended Logic Programs’, *Artif. Intell.*, **109**(1-2), 297–356, (1999).
- [3] G. Brewka, I. Niemelä, and M. Truszcynski, ‘Answer Set Optimization.’, in *IJCAI*, pp. 867–872, (2003).
- [4] G. Brewka, I. Niemelä, and M. Truszcynski, ‘Prioritized Component Systems.’, in *AAAI05*, pp. 596–601, (2005).
- [5] F. Buccafurri, W. Faber, and N. Leone, ‘Disjunctive Logic Programs with Inheritance.’, *Theory and Practice of Log. Program.*, **2**(3), (2002).
- [6] F. Buccafurri, N. Leone, and P. Rullo, ‘Disjunctive Ordered Logic: Semantics and Expressiveness.’, in *Proc. of KR’98*, pp. 418–431, (1998).
- [7] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, ‘Complexity and Expressive Power of Logic Programming.’, *ACM Comput. Surv.*, **33**(3), 374–425, (2001).
- [8] M. Gelfond and V. Lifschitz, ‘Classical Negation in Logic Programs and Disjunctive Databases’, *New Generation Computing*, **9**, (1991).
- [9] M. Gelfond and T. C. Son, ‘Reasoning with Prioritized Defaults.’, in *LPKR*, pp. 164–223, (1997).
- [10] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello, ‘The DLV System for Knowledge Representation and Reasoning’, *ArXiv Computer Science e-prints*, 11004–+, (2002).
- [11] V. Lifschitz, ‘Answer Set Programming and Plan Generation.’, *Artif. Intell.*, **138**(1-2), 39–54, (2002).
- [12] V. Lifschitz, L.R. Tang, and H. Turner, ‘Nested Expressions in Logic Programs’, *Annals of Mathematics and Artif. Intell.*, **25**(3-4), (1999).
- [13] A. Nareyek, E. C. Freuder, R. Fourer, E. Giunchiglia, R. P. Goldman, H. A. Kautz, J. Rintanen, and A. Tate, ‘Constraints and AI Planning.’, *IEEE Intelligent Systems*, **20**(2), 62–72, (2005).
- [14] I. Niemelä and P. Simons, ‘Smodels - an Implementation of the Stable Model and Well-founded Semantics for Normal Logic Programs’, in *Proc. of the 4th LPNMR*, LNCS, pp. 420–429. Springer, (1997).
- [15] C. Sakama and K. Inoue, ‘Prioritized Logic Programming and Its Application to Commonsense Reasoning.’, *Artif. Intell.*, **123**(1-2), (2000).
- [16] T. C. Son, P. H. Tu, M. Gelfond, and A. R. Morales, ‘Conformant Planning for Domains with Constraints-A New Approach.’, in *Proc. of The XXth National Conf. on Artif. Intell. and the 17th Innovative Applications of Artif. Intell. Conf.*, pp. 1211–1216, (2005).

Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems¹

Ph. Chatalic² and G.H. Nguyen³ and M.Ch. Rousset³

Abstract.

In a peer-to-peer inference system, there is no centralized control or hierarchical organization: each peer is equivalent in functionality and cooperates with other peers in order to solve a collective reasoning task. Since peer theories model possibly different viewpoints, even if each local theory is consistent, the global theory may be inconsistent. We exhibit a distributed algorithm detecting inconsistencies in a fully decentralized setting. We provide a fully distributed reasoning algorithm, which computes only *well-founded* consequences of a formula, i.e., with a consistent set of support.

1 Introduction

Recently peer-to-peer (P2P) systems have received considerable attention because their underlying infrastructure is appropriate to scalable and flexible distributed applications over Internet. In P2P systems, there is no centralized control or hierarchical organization: each peer is equivalent in functionality and cooperates with other peers in order to solve a collective task. P2P systems have evolved from simple keyword-based file sharing systems like Napster [2] and Gnutella [1] to semantic data management systems like EDUTELLA [22], PIAZZA [18] or SOMEWHERE [5]. Reasoning in P2P Inference Systems (P2PIS) has been considered very recently (e.g., [3, 12, 4]). The dynamicity of P2PIS imposes to revisit many reasoning problems in order to address them in a decentralized manner. In particular, it is neither feasible to bring all the information to a single server and use standard reasoning algorithms, nor to compute its best partitioning for using partition-based reasoning like in [15, 8].

The local theories of the P2PIS considered in [3, 4] are sets of clauses defined upon sets of propositional variables (the local vocabularies of the peers). A new peer joins an existing P2PIS by establishing *mappings* with other peers, called its *acquaintances*. Mappings are clauses involving variables of distinct peers that state semantic correspondences between different vocabularies. The decentralized algorithm DECA [6] computes the consequences (in some target language) of an input formula w.r.t. the global theory of the P2PIS (i.e. the union of all peer theories). The point is to compute those consequences, without having access to the global theory. DECA is anytime, sound, and complete (under some conditions). It has been implemented in the SOMEWHERE platform and experiments on synthetic data have shown its scalability [4, 6].

This paper focuses on the problem of reasoning with inconsistencies in a P2PIS. Since peer theories model possibly different view-

points, the global theory may be inconsistent even if each local theory is consistent. Given the lack of centralized control, all peers should be treated equally. It would be unfair to refuse the join of a new peer just because the resulting P2PIS becomes inconsistent. Our choice is to accept the presence of inconsistency. The problem is first to *detect* inconsistencies, second to *reason* in spite of them in a satisfactory way. We thus compute only *well-founded* consequences of a formula, i.e., consequences of the formula w.r.t. to a consistent subset of the global theory. Such an approach is not novel in the centralized case but raises new algorithmic issues in the decentralized case because the computation and the storage of *nogoods* (accounting for inconsistencies) are distributed. One has to be able to check the consistency of distributed sets of formulas, w.r.t. distributed sets of nogoods.

We assume each local theory to be consistent. Therefore, the possible inconsistencies result from interactions between local theories and are caused by *mappings*. Before adding a mapping, a peer checks whether this mapping (possibly with other mappings) can be the cause of some inconsistency, i.e., if the empty clause can be produced as one of its consequences. In that case, the peer stores locally as a *nogood* the set of mappings involved in the corresponding proof. At reasoning time, the concerned distributed nogoods must be collected to check whether the proof under construction is well-founded.

In Section 2, we formally define the considered P2PIS. In Section 3, we present a decentralized algorithm for detecting inconsistencies and computing the corresponding nogoods. Section 4 provides a decentralized reasoning algorithm which is well-founded despite possible inconsistencies. In Section 5, we conclude by relating our approach w.r.t. existing work.

2 Peer-to-peer inference systems

The P2PIS that we consider are networks of peer clausal theories $\mathcal{P} = \{P_i\}_{i=1..n}$, such that each peer has a proper *vocabulary* (denoted by \mathcal{V}_{P_i}). We suppose that each peer has a unique identifier (for example, its IP address) and that variable names use in some way this identifier. For simplicity, we just use the index i as the peer identifier and denote a variable A of the peer P_i by A_i . Each local theory of a peer P_i is a set of clauses without duplicated literals. We denote by \mathcal{L}_{P_i} the language of clauses involving only variables of \mathcal{V}_{P_i} .

Definition 1 (Mappings, shared variables and acquaintances)

A mapping is a clause of a peer involving at least a variable of the vocabulary of another peer. A variable of some peer is said to be shared if it appears in a mapping of another peer. The acquaintances of a peer are the peers in the network with which it shares variables.

The global theory $\mathcal{T}(\mathcal{P})$ of a P2PIS $\mathcal{P} = \{P_i\}_{i=1..n}$ is: $\bigcup_{i=1..n} P_i$. Since mappings play a special role in our approach, we distinguish $\mathcal{M} = \bigcup_{i=1..n} M_i$, where M_i is the set of mappings of P_i from

¹This work is supported by France Telecom

²LRI-PCRI - Université Paris-Sud 11, Orsay, France.
 chatalic@lri.fr

³LSR-IMAG - Université Joseph Fourier, Grenoble, France.
 {Gia-Hien.Nguyen,Marie-Christine.Rousset}@imag.fr

$\mathcal{O} = \bigcup_{i=1..n} O_i$, where each O_i is the complementary of M_i . In addition, we assume that each mapping of P_i has a unique identifier prefixed by P_i .

A shared literal is a shared variable or its negation. For a clause c , we denote by $S(c)$ the disjunction of its shared literals and by $L(c)$ the disjunction of its other literals. We suppose that each peer P knows its acquaintances, and given a shared literal l we denote by $ACQ(l, P)$ the set of peers with which P shares the variable of l .

In contrast with other approaches [17, 12], we do not adopt an epistemic or modal semantics for interpreting a P2PIS but we interpret it with the standard semantics of propositional logic. In particular, a variable shared by two peer theories of a given P2PIS is interpreted by the same value in the two peers.

Definition 2 (Semantics of a P2PIS) Let $\mathcal{P} = \{P_i\}_{i=1..n}$ be a P2PIS, an interpretation I of \mathcal{P} is an assignment of the variables of $T(\mathcal{P})$ to true or false. I is a model of a clause c iff one of the literals of c is evaluated to true in I . I is a model of a set of clauses iff it is a model of all the clauses of the set.

- \mathcal{P} is consistent iff $T(\mathcal{P})$ has a model.
- The consequence relation is the standard one: $\mathcal{P} \models c$ iff every model of \mathcal{P} is a model of c . We say that c is an implicate of \mathcal{P} .

Example

Let us consider the P2PIS corresponding to Figure 1. P_1 can be asked by researchers for choosing where to submit their results (demos or papers). For instance, part of its knowledge can model that: $PODS06$ is open for submission; submitting to $PODS06$ entails submitting to $PODS$; only theoretical results are submitted to $PODS$; a demo cannot be submitted to JAIR. P_2 distinguishes proceedings from journals and knows that: submitting to $PODS$ entails submitting to a conference with proceedings; submitting to JAIR entails submitting to a journal; a same result cannot be submitted in a conference and in a journal; patented results cannot be submitted to a journal. P_3 has some knowledge about research valorization policy: software should be patented or presented as demos; theoretical results should be submitted to journals. The knowledge expressed separately by P_1 , P_2 and P_3 using their respective vocabularies can be respectively modelled by the set of clauses O_1 , O_2 and O_3 .

The set M_2 of mappings stored at P_2 states the equivalence between $PODS_1$ and $PODS_2$ (reps. $JAIR_1$ and $JAIR_2$) through the mappings identified by $P_2.1$, $P_2.2$, $P_2.3$ and $P_2.4$. The set M_3 of mappings stored at P_3 also establishes equivalences between variables of P_3 and variables of the two other peers. Note however that mappings clauses do not necessarily result from equivalences.

The reasoning problem

For each peer P_i , we consider a set $\mathcal{TV}_{P_i} \subseteq \mathcal{V}_{P_i}$ of target variables, supposed to represent the variables of interest for the application, (e.g., observable facts in a model-based diagnosis application, or classes storing data in an information integration application). For a set SP of peers of a P2PIS, we define its target language $Target(SP)$ as the language of clauses (including the empty clause) involving only variables of $\bigcup_{P \in SP} \mathcal{TV}_P$.

Definition 3 (P2PIS consequence finding problem)

Given a P2PIS \mathcal{P} , a peer P of \mathcal{P} and a clause $q \in \mathcal{L}_P$, the P2PIS consequence finding problem is to find the set of proper prime implicates of q w.r.t. $T(\mathcal{P})$ that belong to $Target(\mathcal{P})$.

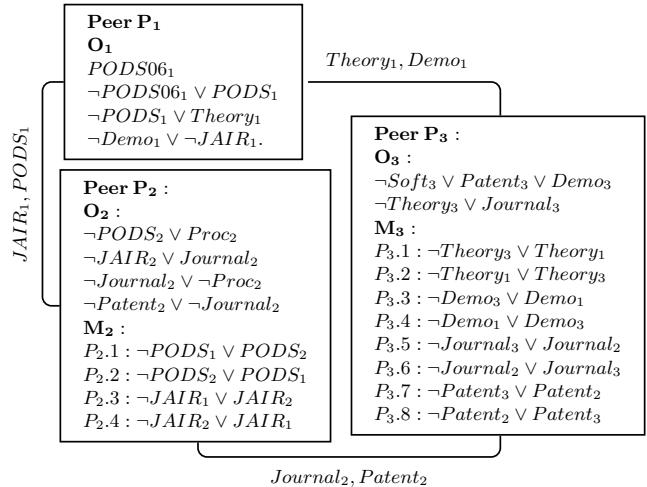


Figure 1. Example P2PIS network (edges labelled by shared variables).

A proper prime implicate of q w.r.t. a (distributed) theory T is a prime implicate of $T \cup \{q\}$, which is not a prime implicate of T .

DECA [3, 6] is the first fully decentralized algorithm being able to solve this problem without having a global view of the system. Details and illustrations of DECA behavior may be found in [6]. Roughly summarized, when running at a peer P and asked to compute the proper prime implicants of a literal q , it first computes all proper prime implicants of this literal w.r.t. the local theory of P , selects those of interest w.r.t. the target language and then split each clause c obtained in this way in two subclauses : $L(c)$ and $S(c)$. $S(c)$ is in turn splitted and for each shared literal l of $S(c)$ DECA asks its appropriate acquaintances (which are running the very same algorithm) to further propagate l in the P2PIS and to return the corresponding results. As soon as consequences are obtained for all literals of $S(c)$ they are recombined together, as well as with $L(c)$, to produce consequences of the splitted clause c . Different branches of reasoning are thus developed throughout the network, following the shared variables, and thus the mappings. In order to ensure termination in presence of cycles as well as to handle the transmission of the results back to the appropriate peers, an *history* is associated to each propagated literal and updated each time that the corresponding reasoning branch goes out from one peer into another peer.

3 P2P detecting inconsistencies and nogoods

As outlined in the introduction, even if each P_i is locally consistent, this is not necessarily the case for $T(\mathcal{P})$. We assume that the causes of inconsistencies are only due to mappings and define a **nogood** ng as a set of mappings such that $\mathcal{O} \cup ng$ is inconsistent. Note that in a nogood ng there necessarily exists some mapping m from which one can derive the empty clause (i.e., with a proof rooted in m). We exploit this property in the decentralized algorithm P2P-NG to detect nogoods accounting for inconsistencies. P2P-NG runs at each peer and is used before adding a new mapping m to a peer P , to check whether the propagation of m into the existing P2PIS produces the empty clause. If that is the case, P stores locally as new nogoods the mappings (including m) involved in the corresponding derivations. We call the set of mappings used in a derivation, its **mapping support**. The P2P-NG algorithm computes the (possibly empty) set of mapping supports of the derivations of the empty clause rooted in

m , starting at the peer P . Since mappings have a unique identifier, mapping supports can efficiently be encoded as sets of mapping identifiers (and similarly for nogoods computed from mapping supports).

P2P-NG is an adaptation of the DECA consequence finding algorithm. It follows the same split-recombine strategy but it has the following significant differences:

- The stopping conditions **must** be changed. The ones in the original DECA algorithm were designed for the computation of *proper prime* implicants only. Here we need to find all the possible ways of deriving the empty clause, we cannot stop the reasoning as soon as we produce the empty clause, or as soon as we find a unit clause in a local peer which is the same as the literal under processing.

- Because we are only looking for \square as a consequence, locally produced consequences c such that $L(c) \neq \square$ can be filtered out.

- While DECA would return $\{\square\}$ as its result if there exists a derivation of the empty clause, P2P-NG returns as many mapping supports as different ways of deriving \square .

We use the following notations:

- For a clause c , $Resolvent+SMS(c, P)$ computes the couples of pairs $(r, SMS(r, P))$ such that r is a local consequence of c w.r.t. P and $SMS(r, P)$ is its corresponding set of local mapping supports.

- For a literal q , \bar{q} denotes its complementary literal.

- A history $hist$ is a sequence of tuples (l, P, c) where l is a literal, P a peer, and c a clause which is a consequence of l on the peer P . A history $[(l_n, P_n, c_n), \dots, (l_1, P_1, c_1), (l_0, P_0, c_0)]$ represents a branch of reasoning initiated by the propagation of the literal l_0 within the peer P_0 , which either has produced locally the clause c_0 in P_0 (in that case, c_0 may have been splitted into its different literals among which l_1 is propagated in P_1), or not (in that case l_0 is simply propagated from P_0 to P_1 and $l_0 = c_0 = l_1$). For every $i \in [0..n-1]$, c_i is a consequence of l_i and P_i , and l_{i+1} is a literal of c_i , which is propagated in P_{i+1} .

- \otimes is the distribution union operator on sets of sets:

$SS_1 \otimes \dots \otimes SS_n = \{S_1 \cup \dots \cup S_n \mid S_1 \in SS_1, \dots, S_n \in SS_n\}$. If $L = \{l_1, \dots, l_p\}$, $\otimes_{l \in L} SS_l$ denotes $SS_{l_1} \otimes \dots \otimes SS_{l_p}$.

Each literal resulting from the splitting of a clause (Line (4) of the P2P-NG algorithm) is processed independently by the P2P-NG $_l$ algorithm. P2P-NG $_l(q, SP, hist)$ checks whether there exists a derivation of \square rooted in the literal q , starting with the computation of local consequences of q , and then recursively following the acquaintances of the visited peers. To ensure termination, it is necessary to keep track of the literals already processed by peers. This is done thanks to $hist$, where $hist$ is the history of the reasoning branch ending up to the propagation of the literal q in SP , which is the set of acquaintances of the last peer added to the history.

Theorem 1 states that the result of P2P-NG(m, P) can be used to characterize nogoods involving the mapping m (by assimilating mappings to their corresponding index).

Theorem 1 Let m be a mapping and P be a peer such that P2P-NG(m, P) $\neq \emptyset$. $\forall ms \in$ P2P-NG(m, P), $ms \cup \{m\}$ is a nogood.

Before adding a new mapping m to its local theory, each peer P first computes P2P-NG(m, P) and stores locally all the nogoods $\{m\} \cup ms$, such that $ms \in$ P2P-NG(m, P) is minimal (for inclusion).

Theorem 2 states that the P2P-NG algorithm is complete, i.e., it enables to find the mapping supports of all the irredundant derivations of the empty clause from a given mapping added to a P2PIS. The proof of Theorem 2 relies on Lemma 1.

Definition 4 (Irredundant derivation) A derivation is irredundant if it does not involve two identical applications of the resolution rule.

Algorithm 1: Detection of the nogoods caused by adding a mapping P2P-NG(m, P)

```

(1) LOCAL( $P$ )  $\leftarrow$  Resolvent+SMS( $m, P$ )
(2) RESULT  $\leftarrow \emptyset$ 
(3) foreach ( $c$  sms)  $\in$  LOCAL( $P$ ) s.t.  $S(c) \neq \square$  and  $L(c) = \square$ 
(4)   foreach literal  $q \in S(c)$ 
(5)     NOGOODS( $q$ )  $\leftarrow$  P2P-NG $_l(q, ACQ(q, P), \emptyset)$ 
(6)     if for every  $q \in S(c)$  NOGOODS( $q$ )  $\neq \emptyset$ 
(7)       UNIONCOMB  $\leftarrow$  sms  $\otimes (\otimes_{q \in S(c)} NOGOODS(q))$ 
(8)     RESULT  $\leftarrow$  RESULT  $\cup$  UNIONCOMB
(9)   return RESULT

```

P2P-NG $_l(q, SP, hist)$

```

(1) if for every  $P \in SP$ ,  $(q, P, \_) \in hist$ 
(2)   return  $\emptyset$ 
(3) else
(4)   SMS( $q$ )  $\leftarrow \{\emptyset\}$ 
(5)   RESULT  $\leftarrow \emptyset$ 
(6)   if  $(\bar{q}, \_, \_) \in hist$ 
(7)     RESULT  $\leftarrow$  RESULT  $\cup \{\emptyset\}$ 
(8)   foreach  $P \in SP$ 
(9)     LOCAL( $P$ )  $\leftarrow \{(q \{\emptyset\})\} \cup$  Resolvent+SMS( $q, P$ )
(10)    RESULT  $\leftarrow$  RESULT  $\cup \bigcup_{P \in SP} SMS(\square, P)$ 
(11)    foreach  $P \in SP$  and ( $c$  sms)  $\in$  LOCAL( $P$ ) s.t.  $S(c) \neq \square$  and  $L(c) = \square$ 
(12)      foreach literal  $l \in S(c)$ 
(13)        SMS( $l$ )  $\leftarrow$  P2P-NG $_l(l, ACQ(l, P), [(q, P, c)|hist])$ 
(14)        if for every  $l \in S(c)$ , SMS( $l$ )  $\neq \emptyset$ 
(15)          UNIONCOMB  $\leftarrow$  sms  $\otimes (\otimes_{l \in S(c)} SMS(l))$ 
(16)          RESULT  $\leftarrow$  RESULT  $\cup$  UNIONCOMB
(17)   return RESULT

```

Theorem 2 Let \mathcal{P} be a P2PIS and m a mapping of a given peer P of \mathcal{P} . Let ms be a mapping support of an irredundant derivation of \square rooted in m . It will be returned by P2P-NG(m, P).

Lemma 1 Let ms be the mapping support of an irredundant derivation of \square rooted in a clause $c: c_1 \vee \dots \vee c_n$ where every c_i is a clause (which can be a unit clause or not) such that there is no literal common to c_i and c_j for $i \neq j$. There exists ms_1, \dots, ms_n where, for every i , ms_i is a mapping support of an irredundant derivation of \square rooted in c_i , such that: $ms = ms_1 \cup \dots \cup ms_n$.

Corollary 1 results directly from Theorem 2. It guarantees that all the minimal nogoods are computed and stored in the P2PIS. It is the key for proving that the reasoning algorithm presented in the next section is well-founded.

Corollary 1 Let $T(\mathcal{P}) = \mathcal{O} \cup \mathcal{M}$ be the global theory of a P2PIS and let S be a set of mappings of \mathcal{M} . If S is a minimal nogood then it is stored at some peer P of \mathcal{P} .

Example (cont.): In the example of Section 2, let us suppose that the different peers join in the following order: P_1 , then P_2 , then P_3 and that their respective mappings are added according to their numbering. At the join of P_2 , the successive adding of the 4 mappings causes no inconsistency. When P_3 joins, the 4 first mappings cause no inconsistency. Let us focus on the 5th one. P2P-NG($\neg Journal_3 \vee Journal_2, P_3$) is triggered where P_3 is the theory containing the clauses of P_3 that are not mappings, and the 4 first mappings (which have been added since they have been checked as not deriving inconsistencies). $\neg Theory_1 \vee Journal_2$ is produced

locally at P_3 as a local consequence of $\neg Journal_3 \vee Journal_2$, with a set of local mapping support equal to $\{\{P_3.2\}\}$. Then, $\neg Theory_1 \vee Journal_2$ is splitted (Line (4) of P2P-NG): $\neg Theory_1$ is processed by P_1 , while $Journal_2$ is processed by P_2 . The propagation of $\neg Theory_1$ produces \square as a local consequence in P_1 with a local set of mapping support equal to $\{\emptyset\}$. Thus $\{\emptyset\}$ is returned to P_3 as the set of mapping supports of the derivation of \square from $Theory_1$. The propagation of $Journal_2$ produces $\neg PODS_1$ as a local consequence in P_2 , with a local set of mapping supports equal to $\{\{P_2.1\}\}$, as well as $\neg Patent_2$ with a local set of mapping support equal to $\{\emptyset\}$. $\neg PODS_1$ is in turn propagated in P_1 , where it produces \square as a local consequence with a local set of mapping supports equal to $\{\emptyset\}$. It is transmitted back to P_2 which, after combination of $\{\emptyset\}$ and $\{\{P_2.1\}\}$ (Line 15) of P2P-NG $_l$ ($Journal_2, \{P_2\}, \emptyset$), transmits back to P_3 the set of mapping supports $\{\{P_2.1\}\}$ for the derivation \square from $Journal_2$. By combination (Line 7) P2P-NG($\neg Journal_3 \vee Journal_2, P_3$) returns $\{\{P_3.2, P_2.1\}\}$. The nogood $\{P_3.5, P_3.2, P_2.1\}$ is thus obtained and stored at P_3 . No other nogood is obtained from the last mappings of P_3 .

4 Peer-to-peer well-founded reasoning

First, we have to define the notion of *well-founded* consequences that can be derived from a given input clause and a possibly inconsistent P2PIS. Many semantics have been proposed and studied for reasoning with (centralized) inconsistent theories ([14] for a survey). We adopt the following one (which is one of the simplest ones), because it makes sense in a decentralized and dynamic setting.

Definition 5 (P2P well-founded implicate) Let \mathcal{P} be an inconsistent P2PIS: r is a well-founded implicate of c w.r.t. \mathcal{P} if r is an implicate of c w.r.t. a consistent subset of $\mathcal{T}(\mathcal{P})$.

The WF-DECA(q, P) algorithm computes well-founded consequences of the (unit) clause q , starting at the peer P . This algorithm extends the original DECA consequence finding algorithm [3, 6] by computing the set of mapping supports of the derivations for each consequence, and by collecting the nogoods encountered during the reasoning. Because of the split/recombination technique used by the algorithm, mapping supports of derivations are only known after the recombination step, and the set of possibly relevant nogoods must be available at this step: if some mapping support includes a nogood, it is discarded; consequences that get an empty set of mapping supports after nogoods filtering are discarded as well.

We use the following notations :

- $LocalConsSSNG(q, P)$ is a local procedure that computes the set of triples $(c \text{ sms } sng)$ such that c is a local consequence of q w.r.t. P , sms is its corresponding set of local mapping supports, and sng is the set of nogoods stored at the peer P that contain a mapping m of some mapping support ms of c .
- \uplus denotes the *merged union* of sets of consequences, i.e. the union of sets of triples of the form $(c \text{ sms } sng)$, where triples corresponding to a same consequence c are merged together, by computing the union of their respective sms and sng .
- \oslash is the distribution union operator on sets of triples of the form $(c \text{ sms } sng)$: $S_1 \oslash \dots \oslash S_n = \{(c_1 \vee \dots \vee c_n \text{ sms}_1 \otimes \dots \otimes \text{sms}_n \text{ sng}_1 \cup \dots \cup \text{sng}_n) / (c_1 \text{ sms}_1 \text{ sng}_1 \in S_1, \dots, (c_n \text{ sms}_n \text{ sng}_n) \in S_n)\}$.

Theorem 3 states that the WF-DECA(q, P) algorithm terminates and returns only well-founded consequences. Its proof relies on showing that each triple $(c \text{ sms } sng)$ returned by WF-

Algorithm 2: Well Founded Distributed Consequence Finding Algorithm

```

WF-DECA( $q, P$ )
(1) WF-DECAH( $q, \{P\}, \emptyset$ )

WF-DECAH( $q, SP, hist$ )
(1) if for every  $P \in SP$ ,  $(q, P, \_) \in hist$ 
(2)   return  $\emptyset$ 
(3) else if  $(\bar{q}, \_, \_) \in hist$ 
(4)   return  $\{\{\square \emptyset \emptyset\}\}$ 
(5) else
(6)   RESULT  $\leftarrow \emptyset$ 
(7)   foreach  $P \in SP$ 
(8)     LOCAL( $P$ )  $\leftarrow \{(q \{\emptyset\} \emptyset)\} \uplus LocalConsSSNG(q, P)$ 
(9)   foreach  $P \in SP$  and  $(c \text{ sms } sng) \in LOCAL(P)$  such that
(10)     $L(c) \in Target(P)$ 
(11)    if  $S(c) \in Target(P)$ 
(12)      RESULT  $\leftarrow$  RESULT  $\uplus \{(c \text{ sms } sng)\}$ 
(13)    if  $S(c) \neq \square$ 
(14)      foreach literal  $l \in S(c)$ 
(15)        ANSWER( $l$ )  $\leftarrow$ 
(16)        WF-DECAH( $l, ACQ(q, P), [(q, P, c)|hist]$ )
(17)    if for every  $l \in S(c)$  ANSWER( $l$ )  $\neq \emptyset$ 
(18)      UNIONCOMB  $\leftarrow \{(L(c) \text{ sms } sng)\} \oslash (\oslash_{l \in S(c)} ANSWER(l))$ 
(19)      foreach  $(c \text{ sms } sng) \in UNIONCOMB$ 
(20)        nsms  $\leftarrow \{ms \in sms / \forall ng \in sng, ng \not\subseteq ms\}$ 
(21)        if  $nsms \neq \emptyset$ 
(22)          RESULT  $\leftarrow$  RESULT  $\uplus \{(c \text{ nsms } sng)\}$ 
(23) return RESULT

```

DECA($q, SP, hist$) is such that either c is a local consequence of q w.r.t. some peer $P \in SP$, or for every $ms \in sms$, $\mathcal{O} \cup ms$ is consistent (using Corollary 1) and c is the result of a derivation rooted in q that only uses clauses from $\mathcal{O} \cup \{l_0, l_1, \dots, l_n, q\} \cup ms$, where l_0, l_1, \dots, l_n are the literals in $hist$.

Theorem 3 Let P be a peer of a P2PIS \mathcal{P} and q a literal belonging to the vocabulary of P . WF-DECA(q, P) terminates and for all triples $(c \text{ sms } sng)$ returned by WF-DECA(q, P), c is a well-founded consequence of q w.r.t. \mathcal{P} .

Example (cont.): Let us illustrate the behaviour of WF-DECA($Soft_3, P_3$), assuming that the only target variables are $PODS_1$ and $JAIR_1$. $Patent_2 \vee Demo_1$ is the only clause produced locally on P_3 with a local part of which (i.e. \square) in $Target(P_3)$. Its local sms is $\{\{P_3.7, P_3.3\}\}$. The only nogood stored at P_3 contains neither $P_3.7$ nor $P_3.3$. The corresponding sng returned by $LocalConsSSNG(Soft, P_3)$ is thus empty. $Patent_2 \vee Demo_1$ is then splitted. When $Patent_2$ is transmitted to P_2 , $\neg JAIR_1$ is the only clause produced locally with a local part (i.e. \square) in $Target(P_2)$. Its local sms is $\{\{P_2.3\}\}$ and its sng is empty. The further propagation of $\neg JAIR_1$ returns an empty result. So the triple $(\neg JAIR_1 \{\{P_2.3\}\} \emptyset)$ is sent back to P_3 as a consequence of $Patent_2$. When $Demo_1$ is transmitted to P_1 the only clause produced locally is $\neg JAIR_1$, which is in $Target(P_1)$. Its local sms is empty, as well as its sng . So the triple $(\neg JAIR_1 \{\emptyset\} \emptyset)$ is sent back to P_3 as a consequence of $Demo_1$. P_3 then combines these two triples obtained from P_2 and P_1 giving $(\neg JAIR_1 \{\{P_2.3\}\} \emptyset)$, which is the only final consequence of $Soft_3$ being in the target language. Since the corresponding sng is empty, it is trivially a well-founded consequence.

5 Conclusion and Related Work

We have presented a fully decentralized approach for reasoning with inconsistencies in propositional P2PIS. Nogoods are discovered each time a new mapping is added. Though these are stored in a completely distributed way, the WF-DECA algorithm guarantees that all consequences it returns are well-founded. For lack of place, many optimization details have been omitted. In particular, calculating *all* possible mapping supports of each consequence is not necessary. Computing only the minimal ones is sufficient. The implementation of this approach is currently under process. An extensive experimental study in the spirit of that of [4, 6] is planned for the near future.

For efficiency reasons our approach for P2P reasoning with more powerful languages is to approximate the peer theories and the queries into propositional logic and to exploit the current infrastructure to compute the corresponding propositional answers. The actual answers are then computed by focusing on the relevant peers.

Reasoning under inconsistency has been widely studied in artificial intelligence, but mainly in the centralized case. There are two kinds of strategies for dealing with inconsistent knowledge bases. The first one is to restore consistency as in belief revision [7, 16]. In the context of peer-to-peer interconnected databases, although not strictly restoring consistency, the work of [9] characterizes consistent answers as those that can be drawn from all minimal repairs of all peer databases and that satisfy as well a set of data exchange constraints (similar to mappings) relating the different peer schemes.

The alternative is just to tolerate inconsistency [10]. In a fully decentralized setting, all peers playing the same role, this seems far more preferable. A wide range of paraconsistent logics have been proposed [20] to avoid the trivialization problem of classical logic. Our approach is closer to coherence systems and argumentative approaches [14], since we consider consequences that can be produced from consistent subsets of $\mathcal{T}(\mathcal{P})$. Since \mathcal{O} is known to be consistent, mapping supports can be considered as justifications supporting the consequences. Our approach is a credulous one, since any subset of \mathcal{M} consistent with \mathcal{O} is considered. Well-founded consequences can also be viewed as either local consequences or as the formulas of some extension of the supernormal default theory [11] $\Delta = (\mathcal{O}, \mathcal{M})$.

The work of [19] on reasoning from inconsistent ontologies is another kind of coherence-based approach. For a given query, it aims at finding a specific consistent subset of the global theory, in which the query (or its negation) classically holds. If it is not possible, the answer is undetermined. The set is constructed by successive consistent expansions, according to some selection function measuring some relevance criterion with the query. They use a syntactical criteria, that selects only formulas sharing some variable with those already selected during the previous iterations. Limiting the choice for the consistent subset clearly reduces the set of accepted consequences.

An epistemic semantics has been proposed in [13] to deal with possible inconsistencies in a P2P Data Inference Systems formalized in a first order multi modal language. It considers the case of local inconsistency (a point not addressed by our approach) as well as global inconsistency. Mapping are formalised in such a way that they cannot be used to propagate information from some locally inconsistent theory. Moreover mappings can only be used to propagate information to a peer, as far as they do not contradict either local information or other non-local information that may be deduced on that peer.

Distributed CSP techniques [24, 23] aim at finding consistent assignments of a set of variables, satisfying a set of distributed constraints. They also propagate nogoods corresponding to invalid partial affectations, that are further stored on the peer that receives them.

This tends to replicate among all agents some global knowledge. This is also the case in distributed ATMS [21]. In comparison, we exploit nogoods only at reasoning time. Only those that may interfere with some mapping support of a related *sms* are transmitted in the *sng*.

REFERENCES

- [1] GNUTELLA. <http://www.gnutella.com>.
- [2] NAPSTER. <http://www.napster.com>.
- [3] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, ‘Distributed reasoning in a peer-to-peer setting’, in *Proc. ECAI’04*, pp. 945–946. IOS Press, (August 2004).
- [4] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, ‘Scalability study of peer-to-peer consequence finding’, in *IJCAI’05*, pp. 351–356. IJCAI, (August 2005).
- [5] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, ‘Somewhere in the semantic web’, in *Proc. of Principles and Practice of Semantic Web Reasoning*, eds., Francois Fages and Sylvain Soliman, volume 3703 of *LNCS*, pp. 1–16. Springer, (Sept. 2005).
- [6] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, ‘Distributed reasoning in a peer-to-peer setting: Application to the semantic web’, *Journal of Artificial Intelligence Research*, **25**, (January 2006).
- [7] Carlos Alchourron, Peter Gardenfors, and David Makinson, ‘On the logic of theory change: Partial meet contraction and revision functions.’, *Journal of Symbolic Logic*, **50**, 510–530, (1985).
- [8] E. Amir and S. McIlraith, ‘Partition-based logical reasoning’, in *Proc. of KR’00*, pp. 389–400, Breckenridge, Colorado, USA, (April 11–15 2000). Morgan Kaufmann Publishers.
- [9] Leopoldo Bertossi and Loreto Bravo, ‘Query answering in peer-to-peer data exchange systems’, *eprint arXiv:cs/0401015*, (January 2004).
- [10] Leopoldo E. Bertossi, Anthony Hunter, and Torsten Schaub, eds. *Inconsistency Tolerance*, volume 3300 of *LNCS*. Springer, 2005.
- [11] Gerhard Brewka, ‘Preferred subtheories: An extended logical framework for default reasoning.’, in *Proc. of IJCAI*, pp. 1043–1048, (1989).
- [12] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, ‘Logical foundations of peer-to-peer data integration’, in *Proc. of PODS’04*, ed., Alin Deutsch, pp. 241–251, Paris, France, (june 14–16 2004). ACM.
- [13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, ‘Inconsistency tolerance in p2p data integration: an epistemic logic approach’, in *Proc. of the 10th Int. Workshop on Database Programming Languages (DBPL 2005)*, (2005).
- [14] Claudette Cayrol, ‘On the relation between argumentation and non-monotonic coherence-based entailment.’, in *Proc. of IJCAI*, pp. 1443–1448, (1995).
- [15] R. Dechter and I. Rish, ‘Directed resolution: the davis-putnam procedure revisited’, in *Proc. of KR’94*, pp. 134–145, Bonn, Germany, (May 24–27 1994). Morgan Kaufmann.
- [16] *Handbook of defeasible reasoning and uncertainty management*, eds., D. Dubois and H. Prade, Kluwer Academic Publishers, 1998.
- [17] Chiara Ghidini and Luciano Serafini, *Frontiers Of Combining Systems* 2, chapter Distributed First Order Logics, 121–139, number 7 in *Studies in Logic and Computation*, Research Studies Press Ltd., 2000.
- [18] Alon Y. Halevy, Zachary Ives, Igor Tatarinov, and Peter Mork, ‘Pi-azza: data management infrastructure for semantic web applications’, in *Proc. of WWW’03*, pp. 556–567. ACM Press, (2003).
- [19] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije, ‘Reasoning with inconsistent ontologies.’, in *IJCAI’05*, pp. 454–459, (2005).
- [20] Anthony Hunter, *Paraconsistent logics*, 11–36, Kluwer Academic Publishers, Norwell, MA, USA, 1998.
- [21] C.L. Mason and R.R. Johnson, *Distributed Artificial Intelligence II*, chapter DATMS: a framework for distributed assumption based reasoning, 293–317, Pitman, 1989.
- [22] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, and al., ‘Edutella: a p2p networking infrastructure based on rdf’, in *Proc. of WWW’02*, pp. 604–615. ACM, (May 2002).
- [23] Marius-Calin Silaghi, Djamila Sam-Haroud, and Boi Faltings, ‘Asynchronous search with aggregations’, in *Proc. of AAAI’00*, pp. 917–922. AAAI Press / The MIT Press, (2000).
- [24] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, ‘The distributed constraint satisfaction problem: Formalization and algorithms’, *IEEE Transactions on Knowledge and Data Engineering*, **10**(5), 673–685, (1998).

Conceptual hierarchies matching: an approach based on discovery of implication rules between concepts

Jérôme David and Fabrice Guillet and Régis Gras and Henri Briand¹

Abstract. Most research works about ontology or schema matching are based on symmetric similarity measures. By transposing the association rules paradigm, we propose to use asymmetric measures in order to enhance matching. We suggest an extensional and asymmetric matching method based on the discovery of significant implications between concepts described in textual documents. We use a probabilistic model of deviation from independence, named implication intensity. Our method is divided into two consecutive stages: (1) the extraction in documents of relevant terms for each concept; (2) the discovery of significant implications between the concepts. Our method is tested on two benchmarks. The results show that some relevant relations, ignored by a similarity-based matching, can be found thanks to our approach.

Keywords: ontology matching, term extraction, implication intensity, extensional matching, association rules.

1 Introduction

With the increase of electronic data and knowledge on the Internet or in companies, the hierarchical categorization of data through ontological forms as taxonomies has been widely used. Web directories such as Yahoo.com and OpenDirectory, the Electronic Document Management, or the Semantic Web with its OWL ontology are examples of such taxonomies.

In the literature, a lot of works deals with schema/ontology matching. The schema or ontology matching aims at finding semantic relations (i.e. equivalence, subsomption, etc) between entities (i.e. concepts, properties) of two schemas/ontologies. These approaches use various techniques such as machine learning [7], FCA-Analysis [17], database schema matching [12], graph matching [13]. These approaches are commonly based on similarity measures for discovering equivalence relations between concepts.

However, the extracted matchings can be enhanced by using asymmetric measures, which deliver more accurate information in the form of implications between concepts. In knowledge discovery in databases (KDD), asymmetric measures, called interestingness measures, are widely used for association rules discovery [1]. Association rules are expressions of the type "if *antecedant* then *consequent*" representing implicative tendencies between conjunctions of attributes in databases. In this paper, we evaluate the use of such asymmetric measures for matching concepts of schemas or ontologies by using the Implication Intensity [9, 2], a probabilistic model of deviation from statistical independence.

Our matching method is both extensional and terminological. It is designed to be used on taxonomies of concepts associated with

textual documents. The idea underlying our approach considers that one concept is more specific than another, if the vocabulary used in the documents associated to the first concept tends to be included in the vocabulary of the other one.

Our method is divided into two consecutive stages: (1) The extraction of concept-relevant terms; (2) The discovery of association rules between concepts.

This paper is organized as follows. In a first section we give an overview of matching approaches. Then, we introduce the Implication Intensity measure, and the concept hierarchy model, and the used formalism. Next, we apply the two stages of our method focusing on rule extraction. Finally, we experiment our method on two datasets and discuss the results obtained.

2 Related works

Many surveys about ontology and schema matching have been proposed in litterature [10], [15], [16]. The two last ones propose a classification and a comparative study of matching approaches. The survey [15] focuses on the database schema matching approaches, while [16] reuses this classification for ontology matching. From these surveys we can distinguish: the extensional approaches (or element-based), and the intensional approaches (or only-schema-based). The matching approaches can be also discriminated regarding the kind of relations that they are based on. Some consider symmetric (equivalence) relations, while other ones also use asymmetric relations such as the subsomption or implication.

The main part of these works propose to process the concept name by using string-similarities (Anchor-PROMPT [14], Cpuid [12], Coma [5], S-MATCH [8]) or/and external oracles such as Wordnet ([8]). They can also use the schema or ontology structure (Similarity Flooding [13], Artemis [3], [5], [14], [12]).

Most of these approaches are intensional and symmetric. None of them are both asymmetric and extensional. Among extensional approaches, we can cite GLUE [6]. This symmetric approach uses Bayesian learners in order to classify instances of the first ontology into the other and vice-versa in order to estimate the joint probability distribution and then predict concept similarities.

We can also notice that there is only one intensional method considering asymmetric relations. The method S-MATCH [8] searches equivalence (=) relation between concepts but also the more general (\sqsupseteq), less general (\sqsubseteq), mismatch (\perp) and overlapping (\sqcap) relations. This method uses a lot of single matchers: 13 linguistic-based matchers and 3 logic-based matchers.

¹ LINA CNRS FRE 2729, Polytechnic School of Nantes University, France, email: jerome.david@polytech.univ-nantes.fr

3 The definition of the Implication Intensity

Let us now consider a finite set T of n individuals described by a set I of p items. Each transaction t can be considered as an itemset so that $t \subseteq I$. $A = \{t \in T; a \subseteq t\}$ is the extension of itemset a and $\bar{B} = T - \{t' \in T; b \subseteq t'\}$ is the extension of \bar{b} . Then, we introduce the quantities $n_a = \text{card}(A)$, $n_{\bar{b}} = \text{card}(\bar{B})$ and $n_{a \wedge \bar{b}} = \text{card}(A \cap \bar{B})$.

An association rule [1] is an implication of the form $a \rightarrow b$, where a and b are disjoined itemsets. In practice, it is quite common to observe a few transactions which contain a and not b without having the general trend to have b when a is present. Therefore, the number $n_{a \wedge \bar{b}}$ of counter-examples must be taken into account to statistically accept to retain or not the rule $a \rightarrow b$.

More precisely, we compare the observed number of counter-examples $n_{a \wedge \bar{b}}$ to a probabilistic model noted $N_{a \wedge \bar{b}}$. Let us assume that we randomly draw two subsets X and Y in T which respectively contain n_a and n_b transactions, i.e. $N_{a \wedge \bar{b}} = \text{card}(X \cup \bar{Y})$.

The implication intensity of the association rule $a \rightarrow b$ is defined by:

$$\varphi(a \rightarrow b) = 1 - \Pr(N_{a \wedge \bar{b}} \leq n_{a \wedge \bar{b}}) \quad (1)$$

The distribution of the random variable $N_{a \wedge \bar{b}}$ depends on the drawing mode [9]. It is established that, under some conditions, the random variable $N_{a \wedge \bar{b}}$ follows a Poisson law with $\lambda = n_a n_{\bar{b}} / n$.

4 Concept hierarchy model and formalisms used

Our approach (figure 1) is designed for conceptual hierarchies of concepts organized by a partial order relation, connected to a set of textual documents.

We define a conceptual hierarchy \mathcal{H} as a quadruplet:

$$\mathcal{H} = (C, \leq, D, \sigma_0) \quad (2)$$

where C is a set of concepts, \leq represents the partial order, D is the set of documents, and σ_0 is the relation which associates a set of documents to each concept (i.e. for a concept $c \in C$, $\sigma_0(c)$ represents the documents associated to c). From the partial order \leq , we extend the relation σ_0 to σ , where:

$$\sigma(c) = \bigcup_{c' \leq c} \sigma_0(c') \quad (3)$$

Then, we transform the hierarchy \mathcal{H} defined on documents in a hierarchy \mathcal{H}' defined on terms as follows:

$$\mathcal{H}' = (C, \leq, T, \gamma_0) \quad (4)$$

where T is the set of relevant terms extracted from D , and $\gamma_0 \subseteq C \times T$ is the relation associating terms to concepts (i.e. $\gamma_0(c)$ represents the set of relevant terms selected for the concept c). Technically, γ_0 is deduced from σ and the relation δ linking terms to documents (i.e. $\delta(t)$ is the set of documents in which the term t appears). From the partial order \leq , we extend the relation γ_0 to γ , where:

$$\gamma(c) = \bigcup_{c' \leq c} \gamma_0(c') \quad (5)$$

The common term set of two hierarchies $\mathcal{H}'_1 = (C_1, \leq_1, T_1, \gamma_1)$ and $\mathcal{H}'_2 = (C_2, \leq_2, T_2, \gamma_2)$ is noted $T_{1 \cap 2} = T_1 \cap T_2$. Next, we define the relation $\gamma_{1 \cap 2}$ which associates a subset of $T_{1 \cap 2}$ for each concept $c \in C_1 \cup C_2$:

$$\gamma_{1 \cap 2}(c) = \begin{cases} \gamma_1(c) \cap T_2 & \text{if } c \in C_1 \\ \gamma_2(c) \cap T_1 & \text{if } c \in C_2 \end{cases} \quad (6)$$

An implicative match set between two hierarchies \mathcal{H}'_1 and \mathcal{H}'_2 is a set of implicative rules. A rule $a \rightarrow b$ between the concepts $a \in C_1$ and $b \in C_2$ represents a quasi-implication from the set of terms $\gamma_{1 \cap 2}(a)$ to the set of terms $\gamma_{1 \cap 2}(b)$.

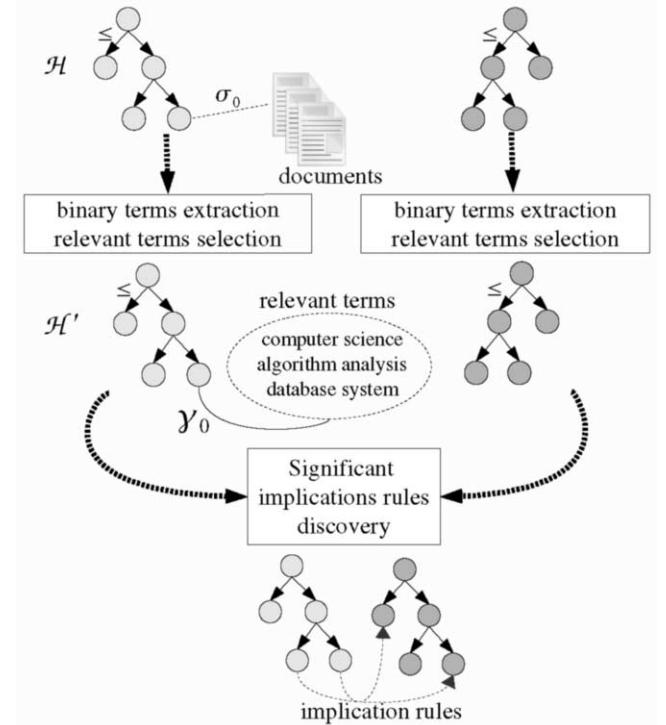


Figure 1. Methodology scheme

5 Extraction and selection of relevant terms

The goal of this process is to extract terms from the documents and then select relevant terms for each concept of the two hierarchies. A term t will be relevant for concept c if t tends to appear in the documents associated with concept c . We choose to associate the term t with concept c if the rule $t \rightarrow c$ has an implication intensity value greater than a given threshold φ_t .

In order to evaluate the rules $t \rightarrow c$, we first extract T_0 , the set of the binary terms (terms composed of two meaningful words) and of the verbs contained in the documents. Binary terms have the advantage to be less ambiguous than simple words. The acquisition of binary terms is performed by software program ACABIT [4]. The textual data are firstly POS-tagged and stemmed by the software program Montylingua [11]. The figure 2 summarizes the term extraction process.

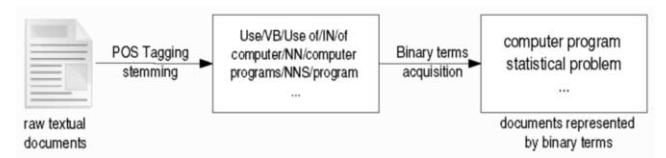


Figure 2. Terms extraction process

Next, we compute all the rules $t \rightarrow c$ and the selection of the relevant terms set of a concept c (noted $\gamma_0(c)$) is defined as follows:

$$\gamma_0(c) = \{t \in T_0 \mid \varphi(t \rightarrow c) \geq \varphi_t\} \quad (7)$$

where φ_t is the implication intensity threshold value and $\varphi(t \rightarrow c)$ is the implication intensity values of the rule $t \rightarrow c$ defined by:

$$\varphi(t \rightarrow c) = 1 - \Pr(N_{t \wedge \bar{c}} \leq n_{t \wedge \bar{c}}) \quad (8)$$

$n_{t \wedge \bar{c}} = \text{card}(\delta(t) - \sigma(c))$ is the observed number of counter-examples, that is to say documents which contain the term t and which are not associated with the concept c . And $N_{t \wedge \bar{c}}$ is the expected number of counter-examples under independence hypothesis.

6 Discovery of significant rules between concepts

6.1 Selection criteria of significant rules

In section 4, we have defined a match result as a set of implication rules between concepts issued from two hierarchies \mathcal{H}_1 and \mathcal{H}_2 . Nevertheless, a lot of rules can be discovered. In this section, we define the implication intensity of a rule between concepts, and then we give two criteria defining the notion of significant rule.

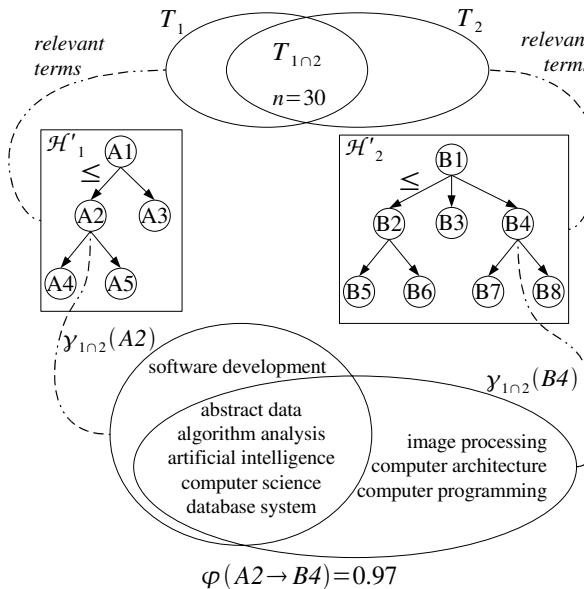


Figure 3. Evaluation of significant rules

The implication intensity of a rule $a \rightarrow b$ (with $a \in C_1$ and $b \in C_2$) is defined by:

$$\varphi(a \rightarrow b) = 1 - \Pr(N_{a \wedge \bar{b}} \leq n_{a \wedge \bar{b}}) \quad (9)$$

where $n_{a \wedge \bar{b}} = \text{card}(\gamma_{1 \cap 2}(a) - \gamma_{1 \cap 2}(b))$ is the number of relevant terms for concept a which are not relevant for concept b . $N_{a \wedge \bar{b}}$ is the expected number of relevant terms for concept a which are not relevant for concept b . On figure 3, the rule $A2 \rightarrow B4$ has $n_{A2 \wedge \bar{B}4} = 1$ counter-examples. Its implication intensity value is:

$$\varphi(A2 \rightarrow B4) = \sum_{k=0}^{n_{A2 \wedge \bar{B}4}} e^{-\lambda} \cdot \frac{\lambda^k}{k!} = 0,97$$

where $\lambda = n_{A2} \cdot n_{\bar{B}4} / n = 6 \times 22 / 30$ (see figure 3).

Thus, the two criteria defining a significant rule are, first, its implication intensity value and, second, the specificity of its consequent combined with the generality of its antecedent. A rule $a \rightarrow b$ (with $a \in C_1$ and $b \in C_2$) will be significant if:

$$\varphi(a \rightarrow b) \geq \varphi_r \quad (10)$$

$$\text{and } \forall x \geq a, \forall y \leq b, \varphi(x \rightarrow y) \leq \varphi(a \rightarrow b) \quad (11)$$

The second criterion (equation 11) selects only generative rules and then permits to reduce redundancy in the extracted rules set. Indeed, from a significant rule $a \rightarrow b$, we can deduce all the rules of the form $u \rightarrow v$ (with $u \leq_1 a$ and $b \leq_2 v$) because at the term level: $\gamma_{1 \cap 2}(b) \subseteq \gamma_{1 \cap 2}(v)$ and $\gamma_{1 \cap 2}(u) \subseteq \gamma_{1 \cap 2}(a)$. We say that the rule $a \rightarrow b$ is generative of the rules set $u \rightarrow v$. For example (figure 3), the rule $A2 \rightarrow B4$ is generative of the rules set $\{A2 \rightarrow B1, A4 \rightarrow B4, A5 \rightarrow B4, A4 \rightarrow B1, A5 \rightarrow B1\}$.

6.2 Algorithms for rule extraction

During the rule extraction step, we can reduce the computation time with the help of the partial order. A top-down search phase enables us to avoid the evaluation of rules having too specific antecedents. This section presents our selection strategy divided into two algorithms.

```

Inputs :
  a : a concept of  $\mathcal{H}_1$ .
Input/Output variables :
   $\mathcal{B}_{current}$  : a set of concepts taking from  $\mathcal{H}_2$ .
  ruleList : the list of selected rules.
Procedure specializeAntecedent( $a, \mathcal{B}_{current}, ruleList$ )
Begin
  ForEach  $b_x \in \mathcal{B}_{current}$  Do
    specializeConsequent( $a, b_x, \mathcal{B}_{current}, 0.0, ruleList$ )
  End Do
  ForEach child  $\in children(a)$  do
     $\mathcal{B}'_{current} := \mathcal{B}_{current}$ 
    specializeAntecedent(child,  $\mathcal{B}'_{current}, ruleList$ )
  End Do
End

```

Figure 4. Algorithm specializing the antecedent

Our first algorithm (figure 4) takes in a concept a from the hierarchy \mathcal{H}_1 and a set of concepts $\mathcal{B}_{current} \subset C_2$ from \mathcal{H}_2 . For each concept of $\mathcal{B}_{current}$, the second algorithm (figure 5) searches and selects valid consequents. It also updates the set $\mathcal{B}_{current}$. And then, this first procedure is recursively launched over the children of a and with a copy of the set $\mathcal{B}_{current}$. The set $\mathcal{B}_{current}$ contains the subtrees of \mathcal{H}_2 with concepts that were selected during the previous recursion steps.

The second algorithm (figure 5) searches a set of valid consequents for the current antecedent a . The search is performed over the set candidate consequents $\{B_x \mid B_x \leq_2 B\}$. A consequent b_s will be selected if the rule $a \rightarrow b_s$ satisfies the two criteria 10 and 11.

This algorithm provides a top-down search of rules in \mathcal{H}_2 , and then explores all branches of the hierarchy. We choose to stop the descent in a branch if $\forall b'_x \leq_2 b_x, \varphi(a \rightarrow b'_x) < \varphi_r$. For a rule $x \rightarrow y$, a property of implication intensity defines $x \cup y$ as the best specialization of the consequent. We exploit this property in order to avoid the evaluation of all rules $a \rightarrow b'_x$.

The describing search method does not consider the roots of hierarchies because all selected terms are associated to root-concepts. The implication intensity value of such rules (i.e. rules which contain root-concepts) is either undefined or equals to 0.

```

Global variable :
 $\varphi_r$  : The Implication Intensity threshold
Inputs :
  a : a concept of  $\mathcal{H}_1$ .
  b : a concept of  $\mathcal{H}_2$ .
   $\varphi_{max}$  : The best value  $\varphi(a \rightarrow b_p)$  with  $b \leq b_p$ 
Input/Output variables :
   $\mathcal{B}_{current}$  : the list of "current" concepts taking from  $\mathcal{H}_2$ .
  ruleList : the list of selected rules.
return value :
  The value  $\varphi$  of the best rule  $a \rightarrow b_x$  with  $b_x \leq b$ 

Function specializeConsequent( $a, b, \mathcal{B}_{current}, \varphi_{max}, ruleList$ )
Begin
  bestChild := FALSE
   $\varphi_{current} := \varphi(a, b)$ 
  returnVal :=  $\varphi_{current}$ 
  If ( $\varphi_{current} < \varphi_r$ ) then
     $\varphi' := \varphi(a, a \cap b)$ 
    If ( $\varphi' < \varphi_r$ ) then
      return  $\varphi_{current}$ 
    EndIf
  EndIf
  ForEach child  $\in children(b)$  do
     $\varphi_{child} := specializeConsequent(a, child, \mathcal{B}_{current}, ruleList)$ 
    If ( $\varphi_{child} \geq \varphi_{current}$ ) then
      bestChild := TRUE
       $\mathcal{B}_{current} := \mathcal{B}_{current} - \{b\}$ 
      If ( $returnVal < \varphi_{child}$ ) then
        returnVal :=  $\varphi_{child}$ 
      EndIf
    EndIf
    If ( $\varphi_{current} > \varphi_r$ ) and  $\neg(bestChild)$  and ( $\varphi_{current} \geq \varphi_{max}$ ) then
      ruleList := ruleList  $\cup (a \rightarrow b)$ 
       $\mathcal{B}_{current} := \mathcal{B}_{current} \cup \{b\}$ 
       $\varphi_{max} := \varphi_{current}$ 
    EndIf
  EndDo
  return returnVal
End

```

Figure 5. Algorithm specializing consequent

7 Experiments

In this section, we choose to compare our method results with a benchmark which is manually matched. We do not confront our results with those provided by other approaches because among the few extensional approaches proposed in the litterature, none are asymmetric. This fact implies that we must "symmetrize" our results before making any experimental comparison with other extensional approaches. This "symmetrization" introduces a strong bias.

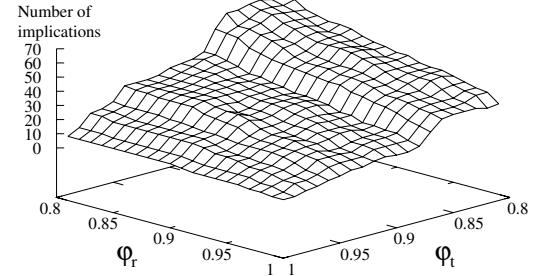
7.1 The analysed data

We experimented our algorithms on two benchmarks proposed in [7]. The first benchmark "Course catalog" describes courses which are proposed at the Cornell and Washington universities. The courses descriptions are hierarchically organised. These two hierarchies contain respectively 166 and 176 concepts to which are associated 4360 and 6957 textual course descriptions. The second benchmark "Company profiles" is issued from the web directories Yahoo.com and Standard.com. These hierarchies describe respectively 13634 and 9504 companies. These company profiles are respectively organised into 115 and 333 sectors and industries.

7.2 Results

We perform both a quantitative test and a qualitative test over the datasets. First, we vary the term threshold φ_t and the rule threshold φ_r in order to analyse their influence over the amount of discovered rules. Secondly, we compare our experimental results with manual matching reference. We vary the thresholds values from 0.8 to 1 for the two tests.

Number of implications between Cornell and Washington course catalogs



Number of implications between Yahoo.com and Standard.com directories

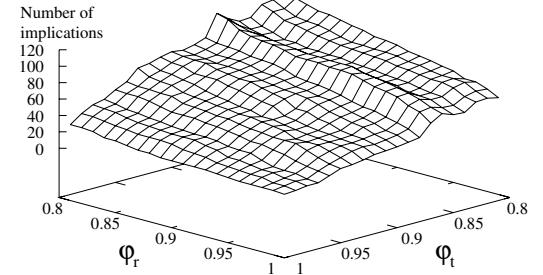


Figure 6. Influence of threshold values over the amount of selected rules

The two graphs of figure 6 shows that the terms selection threshold φ_t has a greater influence than the rule selection threshold φ_r . For example, a φ_t increase of 0.1 unit entails a decrease of 2.15 rules while the same increase of 0.1 unit for φ_r only entails a decrease of 1.2 rules.

Secondly, we perform the qualitative test of precision and recall with the "course catalog" dataset. We compare the results produced by our approach with a reference matching pair set provided by [7]. However their relations are symmetric while ours are asymmetric. Consequently we only retain equivalence relations from ours results (If $a \rightarrow b$ and $b \rightarrow a$ then $a \leftrightarrow b$). Finally, we built the two following graphs respectively representing the evolution of the precision value and the evolution of the recall value in function of chosen threshold values φ_t and φ_r . These two measures issued from information retrieval are defined as follows: let us consider F the set of matching pairs found using our approach and R the set of "reference" matching pairs. The precision ($precision = card(F \cap R)/card(F)$) measures the share of real matching pairs among all found ones. The recall ($recall = card(F \cap R)/card(R)$) measures the share of real correspondances that is found.

Figure 7 shows that the term selection threshold φ_t has a greater influence than the rule selection threshold φ_r . We obtain good precision values (from 0.71 to 1). Nevertheless, the recall values are quite bad: we notice an average recall value equals to 0.29. The best value is equal to 0.54. Our method seems to be too selective.

We found two arguments explaining these recall results. The former is related to the term selection phase. Indeed, a lot of leaves-concepts of the hierarchies cannot be compared to other concepts because they do not have relevant terms selected. Leaves-concepts are not associated with a lot of documents, so it is difficult to assign relevant terms to those concepts. The latter reason, is due to the kind of studied relations. We consider in our approach implicative relations from which we deduce equivalence relations. For example, if

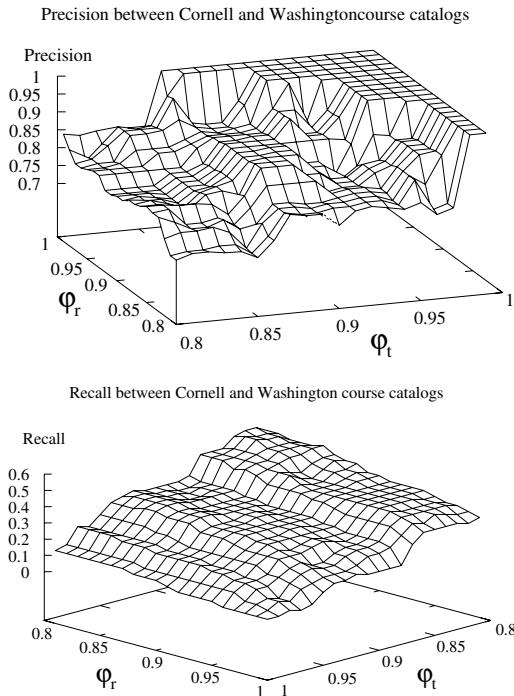


Figure 7. Thresholds values influence over the precision and the recall

we hold the term selection threshold φ_t to 0.83 and the rule selection threshold φ_r to 0.82, we obtain 24 false positives. Among them, 6 are considered as simple implications and not equivalence.

However, our method discovered some meaningful rules which are not in the manual matching reference. For example, we have obtain : From Cornell to Washington :

City and Regional Planning → Urban Planning URBDP
Cognitive Studies Program → Psychology PSYCH

Department of Aerospace Studies → Aerospace Studies A S
Electrical and Computer Engineering → Electrical Engineering E E

From Washington to Cornell :

Atmospheric Sciences ATM S → Earth and Atmospheric Sciences

Our approach is not sensitive to concept name: it led to the following rule "Cognitive Studies Program → Psychology PSYCH" that would not be found with an approach based on string similarity. The selection of relevant terms for concepts permits to take into account the semantics of the concepts.

8 Conclusion

In this paper, we proposed an extensional matching method based on the discovery of significant implication rules between concepts. Our approach takes in two hierarchies of concepts to be matched and the textual corpus indexed to these concepts. The matching task is divided into two stages: (1) the extraction and selection of relevant terms for each concepts; (2) the discovery of significant rules between concepts by using their relevant terms set. The main advantages of this method are the consideration of semantic by using binary terms contained in the corpus and the discovery of rules allowing to enhance the produced matching results only regarding

similarity-based matching systems. We implemented and tested our algorithms with two real catalogs respectively related to company profiles and course catalogs. The results show that we distinguish implication and equivalence relations. We can also notice that our prototype has found several relevant relations not considered by the manual reference matching pair set.

In our point of view, and after a study of related works, our method seems really novel and unique because: (1) it works on terminological level and not on the document level, (2) it is based on asymmetric relations (which offer a stronger semantic for user), (3) it operates an original reduction of rule redundancies.

Currently, we propose an extensional individual matcher. In the near future, we will propose a schema based matcher and combine the two approaches in order to enhance the ontology matching task.

REFERENCES

- [1] R. Agrawal, T. Imielinski, and A. Swami, 'Mining association rules between sets of items in large databases', in *the 1993 ACM SIGMOD international conference on Management of data*, pp. 207–216. ACM Press, (1993).
- [2] J. Blanchard, P. Kuntz, F. Guillet, and R. Gras, *Implication intensity: from the basic statistical definition to the entropic version*, chapter 28, 473–485, CRC Press, 2003.
- [3] S. Castano, V. De Antonellis, and S. De Capitani Di Vimercati, 'Global viewing of heterogeneous data sources', *IEEE Transactions on Knowledge and Data Engineering*, **13**(2), (2001).
- [4] B. Daille, 'Conceptual structuring through term variations', in *ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, eds., F. Bond, A. Korhonen, D. MacCarthy, and A. Villavicencio, pp. 9–16, (2003).
- [5] H.H. Do and E. Rahm, 'Coma - a system for flexible combination of schema matching approaches', in *the International Conference on Very Large Data Bases (VLDB '02)*, pp. 610–621, (2002).
- [6] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, 'Learning to map between ontologies on the semantic web', in *The Eleventh International WWW Conference*, pp. 662–673. ACM Press, (2002).
- [7] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, *Ontology Matching : a machine learning approach*, 397–416, Springer-Velag, 2004.
- [8] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, 'S-match: an algorithm and an implementation of semantic matching', in *European Semantic Web Symposium*, LNCS 3053, pp. 61–75, (2004).
- [9] R. Gras et al., *L'implication statistique, une nouvelle méthode exploratoire de données*, La pensée sauvage, 1996.
- [10] Y. Kalfoglou and M. Schorlemmer, 'Ontology mapping: the state of the art', *Knowledge Engineering Review*, **18**(1), 1–31, (2003).
- [11] H. Liu. Montylingua: An end-to-end natural language processor with common sense, 2004.
- [12] J. Madhavan, P. A. Bernstein, and E. Rahm, 'Generic schema matching with cupid', in *the International Conference on Very Large Data Bases (VLDB'01)*, pp. 49–58, (2001).
- [13] S. Melnik, H. Garcia-Molina, and E. Rahm, 'Similarity flooding: A versatile graph matching algorithm and its application to schema matching', in *the 18th International Conference on Data Engineering (ICDE'02)*, pp. 117–128. IEEE Computer Society, (2002).
- [14] N. Noy and M. Musen, 'Anchor-prompt: Using non-local context for semantic matching', in *the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 63–70, (2001).
- [15] E. Rahm and P. A. Bernstein, 'A survey of approaches to automatic schema matching', *The VLDB Journal*, **10**(4), 334–350, (2001).
- [16] P. Shvaiko and J. Euzenat, 'A survey of schema-based matching approaches', *Journal on Data Semantics IV*, **4**(LNCS 3730), 146–171, (2005).
- [17] G. Stumme and A. Maedche, 'FCA-MERGE: Bottom-up merging of ontologies', in *IJCAI*, pp. 225–234, (2001).

CTL Model Update: Semantics, Computations and Implementation

Yulin Ding and Yan Zhang¹

Abstract. Minimal change is a fundamental principle for modeling system dynamics. In this paper, we study the issue of minimal change for Computational Tree Logic (CTL) model update. We first propose five primitive operations which capture the basic update of the CTL model, and then define the minimal change criteria for CTL model update based on these primitive operations. We provide essential semantic and computational characterizations for our CTL model update approach. We develop a formal algorithm to implement this update that employs the underlying minimal change principle. We also present a CTL model update example using the well known microwave oven scenario.

1 Introduction

Over the past decade, automated formal verification tools, such as model checkers, have shown their ability to provide a thorough automatic error diagnosis in complex designs, e.g. [10]. The current state of the art model checkers, such as SMV [4], NuSMV [3] and Cadence SMV [9], employ SMV specification language for both CTL and Linear Temporal Logic (LTL) model checking. Other model checkers such as SPIN [7] use Promela specification language for on the fly LTL model checking. Also, the MCK [5] model checker was developed by integrating a knowledge operator to verify the knowledge related properties in security protocols.

Along with model checking, error repair has begun to employ a formal methods approach. Buccafurri et al. [2] used abductive model revision techniques to repair errors in concurrent programs. They aimed at using techniques and concepts from model checking by combining them with AI principles. In the paper of Harris and Ryan [6], model checking is formalized with a belief updating operator to satisfy classical proposition knowledge update KM postulates U1-U8. Baral and Zhang [1] presented a formal approach of knowledge update based on single agent S5 Kripke structures. As they argued, their approach of knowledge update could be integrated with model checking technology towards a more general automatic system modification. In this paper, we consider the problem of CTL model update from both theoretical and implementational perspectives. We first consider five primitive CTL model update operations, and based on these operations, we define a minimal change principle for CTL model update. We then investigate the essential semantic and computational properties of CTL model update. Based on these findings, we develop an algorithm to perform CTL model update. By presenting a case study, we also show how our system prototype is applied for system modification.

¹ Intelligent Systems Laboratory, School of Computing and Mathematics, University of Western Sydney, Australia. Email: {yding,yan}@scm.uws.edu.au. The corresponding author: Yan Zhang.

2 CTL Syntax and Semantics: An Overview

To begin with, we briefly review the syntax and semantics of CTL. Readers are referred to [4] and [8] for details.

Definition 1 [4] Let AP be a set of atomic propositions. A Kripke model M over AP is a three tuple $M = (S, R, L)$ where

1. S is a finite set of states,
2. $R \subseteq S \times S$ is a transition relation,
3. $L : S \rightarrow 2^{AP}$ is a function that assigns each state with a set of atomic propositions.

Definition 2 [8] Computation tree logic (CTL) has the following syntax given in Backus naur form:

$$\begin{aligned} \phi ::= & \top \mid \perp \mid p \mid (\neg\phi) \mid (\phi \wedge \psi) \mid (\phi \vee \psi) \mid \phi \supset \psi \mid AX\phi \mid EX\phi \\ & \mid AG\phi \mid EG\phi \mid AF\phi \mid EF\phi \mid A[\phi \cup \psi] \mid E[\phi \cup \psi] \end{aligned}$$

where p is any propositional atom.

A CTL formula is evaluated on a Kripke model M . A path in M from a state s is an infinite sequence of states $\pi \stackrel{\text{def}}{=} [s_0, s_1, \dots, s_{i-1}, s_i, s_{i+1}, \dots]$ such that $s_0 = s$ and $(s_i, s_{i+1}) \in R$ holds for all $i \geq 0$. We write $(s_i, s_{i+1}) \subseteq \pi$ and $s_i \in \pi$. If we express a path as $\pi = [s_0, s_1, \dots, s_i, \dots, s_j, \dots]$ and $i < j$, we say that s_i is a state *earlier* than s_j in π as $s_i < s_j$. For simplicity, we may use $\text{succ}(s)$ to denote state s' if there is a relation (s, s') in R .

Definition 3 [8] Let $M = (S, R, L)$ be a Kripke model for CTL. Given any s in S , we define whether a CTL formula ϕ holds in state s . We denote this by $(M, s) \models \phi$. The satisfaction relation \models is defined by structural induction on all CTL formulas:

1. $(M, s) \models \top$ and $M, s \not\models \perp$ for all $s \in S$.
2. $(M, s) \models p$ iff $p \in L(s)$.
3. $(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$.
4. $(M, s) \models \phi_1 \wedge \phi_2$ iff $(M, s) \models \phi_1$ and $(M, s) \models \phi_2$.
5. $(M, s) \models \phi_1 \vee \phi_2$ iff $(M, s) \models \phi_1$ or $(M, s) \models \phi_2$.
6. $(M, s) \models \phi_1 \rightarrow \phi_2$ iff $(M, s) \not\models \phi_1$, or $(M, s) \models \phi_2$.
7. $(M, s) \models AX\phi$ iff for all s_1 such that $(s, s_1) \in R$, $(M, s_1) \models \phi$.
8. $(M, s) \models EX\phi$ iff for some s_1 such that $s \rightarrow s_1$, $(M, s_1) \models \phi$.
9. $(M, s) \models AG\phi$ holds iff for all paths $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, and all s_i along the path, $(M, s_i) \models \phi$.
10. $(M, s) \models EG\phi$ holds iff there is a path $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, and for all s_i along the path, $(M, s_i) \models \phi$.
11. $(M, s) \models AF\phi$ holds iff for all paths $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, there is some s_i in the path such that $(M, s_i) \models \phi$.
12. $(M, s) \models EF\phi$ holds iff there is a path $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, and for some s_i along the path, $(M, s_i) \models \phi$.

13. $(M, s) \models A[\phi_1 \cup \phi_2]$ holds iff for all paths $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, the path satisfies $\phi_1 \cup \phi_2$, i.e. there is some s_i along the path, such that $(M, s_i) \models \phi_2$, and, for each $j < i$, $(M, s_j) \models \phi_1$.
 14. $(M, s) \models E[\phi_1 \cup \phi_2]$ holds iff there is a path $[s_0, s_1, s_2, \dots]$, where $s_0 = s$, the path satisfies $\phi_1 \cup \phi_2$, i.e. there is some s_i along the path, such that $(M, s_i) \models \phi_2$, and, for each $j < i$, $(M, s_j) \models \phi_1$.

In the rest of this paper, without explicit declaration, we will assume that all CTL formulas occurring in our context will be satisfiable. For instance, when we consider to update a Kripke model with a CTL formula ϕ , we already assume that ϕ is satisfiable.

3 Minimal Change for CTL Model Update

Given a CTL kripke model and a (satisfiable) CTL formula, we consider how this model can be updated in order to satisfy the given formula. We first give the following general definition on CTL model update.

Definition 4 (CTL Model Update) Given a CTL Kripke model $M = (S, R, L)$ and a CTL formula ϕ . An update of $M = (M, s_0)$ where $s_0 \in S$ with ϕ is a CTL Kripke model $M' = (S', R', L')$ such that $M' = (M', s'_0) \models \phi$ where $s'_0 \in S'$. We use $\text{Update}(M, \phi)$ to denote the result M' and $\text{Update}(M, \phi) = M$ if $M \models \phi$.

Definition 4 only presents an essential requirement for a CTL model update and does not tell how such update should be conducted. Basically, as in traditional knowledge base update [11], we would expect that a CTL model update obeys an underlying minimal change principle. Furthermore, this minimal change should be defined based on some operational process so that a concrete algorithm for CTL model update can be implemented. To this end, we first consider five primitive operations on the CTL model that provide a basis for all complex CTL model updates.

3.1 Primitive Operations

The operations to update the CTL model can be decomposed into five types identified as PU1, PU2, PU3, PU4 and PU5. These primitive updates are defined in their simplest forms as follows.

PU1: Adding a relation only

Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of M having only added one new relation. That is $S' = S$, $L' = L$, and $R' = R \cup \{(s_{ar}, s_{ar2})\}$ where $(s_{ar}, s_{ar2}) \notin R$ for one pair of $s_{ar}, s_{ar2} \in S$.

PU2: Removing a relation only

Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of M having only removed one existing relation. That is, $S' = S$; $L' = L$, and $R' = R - \{(s_{rr}, s_{rr2})\}$ where $(s_{rr}, s_{rr2}) \in R$ for one pair of $s_{rr}, s_{rr2} \in S$.

PU3: Substituting a state and its associated relation(s) only

Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of M having only substituted one existing state and its associated relation(s). That is, $S' = S[s/s_{ss}]$ (i.e. S' is the set of states where one state s in S is substituted by s_{ss}), $R' = R \cup \{(s_i, s_{ss}), (s_{ss}, s_j) \mid (s_i, s), (s, s_j) \in R\} - \{(s_i, s), (s, s_j) \mid (s_i, s), (s, s_j) \in R\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ and $L'(s_{ss}) = \tau(s_{ss})$, where τ is a truth assignment on s_{ss} .

PU4: Adding a state and its associated relation(s) only

Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the

result of M having only added one new state and its associated relation(s). That is, $S' = S \cup \{s_{as}\}$, $R' = R \cup \{(s_i, s_{as}), (s_{as}, s_j) \mid$ for some $s_i, s_j \in S \cap S'\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ and $L'(s_{as}) = \tau(s_{as})$, where τ is the truth assignment on s_{as} .

PU5: Removing a state and its associated relation(s) only

Given $M = (S, R, L)$, its updated model $M' = (S', R', L')$ is the result of M having only removed one existing state and its associated relation(s). That is, $S' = S - \{s_{rs} \mid s_{rs} \in S\}$, $R' = R - \{(s_i, s_{rs}), (s_{rs}, s_j) \mid$ for some $s_i, s_j \in S\}$, and $L'(s) = L(s)$ for all $s \in S \cap S'$ (note $S' \subset S$).

We call the above five operations *atomic* since all changes on a CTL model can be expressed by these five operations. It may be argued that PU3 can be expressed by PU4 and PU5. However, we will treat state substitution differently from a combination of state addition and state deletion. That is, in our context, whenever a state substitution is needed, we will apply PU3 directly rather than PU4 followed by PU5. This will simplify our definition on CTL model minimal change (see next).

3.2 Defining Minimal Change

In order to define the minimal change criteria for CTL model update, we need to consider changes on both states and relations for the underlying CTL models. We achieve this by specifying the differences of states and relations on CTL models through primitive operations. First we introduce some useful notions.

Given any two sets X and Y , the *symmetric difference* between X and Y is denoted as $\text{Diff}(X, Y) = (X - Y) \cup (Y - X)$. Given two CTL models $M = (S, R, L)$ and $M' = (S', R', L')$, for each primitive operation PU_i ($i = 1, \dots, 5$), $\text{Diff}_{\text{PU}_i}(M, M')$ denotes the differences between two CTL models where M' is a resulting model from M , that make clear that several operations of this type may occur. Since PU1 and PU2 only change relations, we define $\text{Diff}_{\text{PU}_i}(M, M') = (R - R') \cup (R' - R)$ ($i = 1, 2$). For operations PU3, PU4 and PU5, on the other hand, we define $\text{Diff}_{\text{PU}_i}(M, M') = (S - S') \cup (S' - S)$ ($i = 3, 4, 5$). Although any changes on states caused by PU3-PU5 will also imply corresponding changes on relations, we only count changes on states and take state changes as the primitive factor to measure the difference between M and M' ². For operation PU3, we should also consider the case that when a state is substituted by a new state, we require the difference between these two states to be minimal under the condition of satisfying the updated formula. Finally, we specify

$$\text{Diff}(M, M') = (\text{Diff}_{\text{PU}_1}(M, M'), \dots, \text{Diff}_{\text{PU}_5}(M, M')).$$

Let M , M_1 and M_2 be three CTL models. We denote $\text{Diff}(M, M_1) \preceq \text{Diff}(M, M_2)$ iff (1) for each i ($i = 1, \dots, 5$), $\text{Diff}_{\text{PU}_i}(M, M_1) \subseteq \text{Diff}_{\text{PU}_i}(M, M_2)$; or (2) $\text{Diff}_{\text{PU}_i}(M, M_1) \subseteq \text{Diff}_{\text{PU}_i}(M, M_2)$ for $i = 1, 2, 4, 5$, and $|\text{Diff}_{\text{PU}_3}(M, M_1)| = |\text{Diff}_{\text{PU}_3}(M, M_2)|$ implies for each state s in M substituted by s_1 and s_2 in M_1 and M_2 respectively, $\text{Diff}(s, s_1) \subseteq \text{Diff}(s, s_2)$.

Definition 5 (Closeness Ordering) Given three CTL Kripke models M , M_1 and M_2 , where M_1 and M_2 are obtained from M by applying PU1-PU5 operations. We say that M_1 is closer or as close to M as M_2 , denoted as $M_1 \leq_M M_2$, iff $\text{Diff}(M, M_1) \preceq \text{Diff}(M, M_2)$.

² In our full paper, we justify this principle by showing that as long as a state change is fixed, associated relation changes do not play a crucial role to influence the resulting CTL model.

$\text{Diff}(M, M_2)$. We denote $M_1 <_M M_2$ if $M_1 \leq_M M_2$ and $M_2 \not\leq_M M_1$.

Definition 6 (Admissible Update) Given a CTL Kripke model $M = (S, R, L)$, $\mathcal{M} = (M, s_0)$ where $s_0 \in S$, and a CTL formula ϕ , $\text{Update}(\mathcal{M}, \phi)$ is called admissible if the following conditions hold:
(1) $\text{Update}(\mathcal{M}, \phi) = (M', s'_0) \models \phi$ where $M' = (S', R', L')$ and $s'_0 \in S'$; and (2) there does not exist another resulting model $M'' = (S'', R'', L'')$ and $s''_0 \in S''$ such that $(M'', s''_0) \models \phi$ and $M'' <_M M'$.

4 Semantic Characterizations

From Definition 6, we observe that for a given CTL Kripke model M and a formula ϕ , there may be many admissible updates to satisfy ϕ , where some updates are simpler than others. In this section, we provide various semantic characterizations on CTL model update that present possible solutions to achieve admissible updates under certain conditions. In general, in order to achieve admissible update results, we may have to combine various primitive operations during an update process. Nevertheless, as will be shown in the following, for many situations, a single type primitive operation will be enough to achieve an admissible updated model. These characterizations also play an essential role to simplify CTL model update implementations.

Theorem 1 Let $M = (S, R, L)$ be a Kripke model and s_0 be an initial state in S and $\mathcal{M} = (M, s_0) \not\models EX\phi$, where ϕ is a propositional formula. Then an admissible updated model $\mathcal{M}' = \text{Update}(\mathcal{M}, EX\phi)$ can be obtained by doing one of the following operations:

1. PU3 is applied to any $\text{succ}(s_0)$ once to substitute it with a new state $s^* \models \phi$ and $\text{Diff}(\text{succ}(s_0), s^*)$ to be minimal, or PU4 is applied one time after adding a new state $s^* \models \phi$ and a new relation (s_0, s^*) ;
2. if there exists some $s_i \in S$ such that $s_i \models \phi$ and $s_i \neq \text{succ}(s_0)$, PUI is applied one time to add a new relation (s_0, s_i) .

Theorem 1 provides two cases where admissible CTL model update results can be achieved for formula $EX\phi$. The first case says that we can either select one of s_0 's successor states and substitute it with a new state satisfying ϕ (i.e. applying PU3 one time), or simply add a new state that satisfies ϕ as a successor of s_0 (i.e. applying PU4 one time). The second case indicates that if there is some state s_i in S that already satisfies ϕ , then it is enough to simply add a new relation (s_0, s_i) to make it as a successor of s_0 . It is easy to see that both cases will yield new CTL models that satisfy $EX\phi$. The theorem shows that such new models are also minimal with respect to the original CTL model.

Theorem 2 Let $M = (S, R, L)$ be a Kripke model and $\mathcal{M} = (M, s_0) \not\models AG\phi$, where $s_0 \in S$ and ϕ is a propositional formula. Then an admissible updated model $\mathcal{M}' = \text{Update}(\mathcal{M}, AG\phi)$ can be obtained by the following: for each path starting from s_0 : $\pi = [s_0, \dots, s_i, \dots]$:

1. if for all $s < s_i$ in π , $s \models \phi$ but $s_i \not\models \phi$, PU2 is applied to remove relation (s_{i-1}, s_i) , or PU5 is applied to remove s_i and its associated relations;
2. PU3 is applied to all states s in π not satisfying ϕ to substitute s with $s^* \models \phi$ and $\text{Diff}(s, s^*)$ to be minimal.

In Theorem 2, Case 1 considers a special form of path π where the first i states starting from s_0 already satisfy formula ϕ . Under this situation, we can simply cut off the path (i.e. applying PU2 or PU5 one time) to disconnect all other states not satisfying ϕ . Case 2 is straightforward.

Theorem 3 Let $M = (S, R, L)$ be a Kripke model, $\mathcal{M} = (M, s_0) \not\models EG\phi$, where $s_0 \in S$ and ϕ is a propositional formula. Then an admissible updated model $\mathcal{M}' = \text{Update}(\mathcal{M}, EG\phi)$ can be obtained by the following: Select a path $\pi = [s_0, s_1, \dots]$ from M which contains minimal number of states not satisfying ϕ , and then

1. if for all $s' \in \pi$ such that $s' \not\models \phi$, there exist $s_i, s_j \in \pi$ satisfying $s_i < s' < s_j$ and $s_i \models \phi$ and $s_j \models \phi$, then PUI is applied to add a relation (s_i, s_j) , or PU4 is applied to add a state $s^* \models \phi$ and new relations (s_i, s^*) and (s^*, s_j) ;
2. if there exists some $s_i \in \pi$ ($i > 1$) such that for all $s' < s_i$, $s' \models \phi$ and $s_i \not\models \phi$, then PU2 is applied to remove relation (s_{i-1}, s_i) , or PU5 is applied to remove state s_i and its associated relations;
3. for all $s' \in \pi$, $s' \not\models \phi$, then PU3 is applied to substitute all s' with new state $s^* \models \phi$ and $\text{Diff}(s, s^*)$ to be minimal.

Proof: We prove case 1 here while proofs for the other two cases are presented in our full paper. Without loss of generality, we assume for the selected path π , there exists one state s' that does not satisfy ϕ , and all other states in π satisfy ϕ . We also assume that such s' is in the middle of path π . Therefore, there are two other states s_i, s_j in π such that $s_i < s' < s_j$. That is, $\pi = [s_0, \dots, s_{i-1}, s_i, \dots, s', \dots, s_j, s_{j+1}, \dots]$. We first consider to apply PU1. It is clear that by applying PU1 to add a new relation (s_i, s_j) , a new path is formed: $\pi' = [s_0, \dots, s_{i-1}, s_i, s_j, s_{j+1}, \dots]$. Note that each state in π' is also in path π and $s' \notin \pi'$. According, we know $EG\phi$ is held in the new model $M' = (S, R \cup \{(s_i, s_j)\}, L)$ at state s_0 . On the other hand, we consider $\text{Diff}(M, M')$. Clearly, $\text{Diff}(M, M') = (\{(s_i, s_j)\}, \emptyset, \emptyset, \emptyset, \emptyset)$, which implies M' must be a minimal model with respect to \leq_M that satisfies $EG\phi$.

Now we consider to apply PU4. In this case, we will have a new model $M' = (S \cup \{s^*\}, R \cup \{(s_i, s^*), (s^*, s_j)\}, L')$ where L' is an extension of L on new state s^* that satisfies ϕ . We can see that $\pi' = [s_0, \dots, s_i, s^*, s_j, \dots]$ is a path in M' which shares all states with path π except the state s' in π and those states between s_{i+1} and s_{j-1} including s' in π . So we also have $(M', s_0) \models EG\phi$. On the other hand, we have $\text{Diff}(M, M') = (\emptyset, \emptyset, \emptyset, \{s^*\}, \emptyset)$. Obviously, M' is a minimal model with respect to \leq_M that satisfies $EG\phi$. \square

Theorem 3 characterizes three typical situations for the update with formula $EG\phi$. Basically, this theorem says that in order to make formula $EG\phi$ be true, we first select a path, then we can either make a new path based on this path so that all states in the new path satisfy ϕ (i.e. Case 1), trim the path from the state where all previous states satisfy ϕ (i.e. Case 2); or simply substitute all states not satisfying ϕ in the path with new states satisfying ϕ (i.e. Case 3). Our proof shows that resulting models from these operations are admissible.

In our full paper, we also provide semantic characterizations for updates with other CTL formulas such as $EF\phi$, $AX\phi$, $AF\phi$, $E[\phi \cup \psi]$ and $A[\phi \cup \psi]$. All these results have been proved to be useful in simplifying the implementation and improving the efficiency of the CTL model update performance.

5 Computational Properties

In this section, we study the computational complexity of our approach for CTL model update. We first present the following general result.

Theorem 4 *Given two CTL Kripke models $M = (S, R, L)$ and $M' = (S', R', L')$, where $s_0 \in S$ and $s'_0 \in S'$, and a CTL formula ϕ . Deciding whether (M', s'_0) is an admissible result of $\text{Update}((M, s_0), \phi)$ is co-NP-complete. The hardness holds even if ϕ is of the form $\text{EX}\psi$ where ψ is a propositional formula.*

Proof: Membership proof. First from [4], we know that checking whether $(M', s'_0) \models \phi$ can be done in time $\mathcal{O}(|\phi| \cdot (|S| + |R|))$. In order to check whether (M', s'_0) is an admissible update result, we need to check whether M' is minimal with respect to ordering \leq_M . To do so, we consider the complement of the problem. That is to check whether M' is not a minimal model. Therefore, we do two things: (1) guessing a CTL model $M'' = (S'', R'', L'')$ satisfying ϕ for some $s'' \in S''$; and (2) testing whether $M'' <_M M'$. Step (1) can be done in polynomial time. To check $M'' <_M M'$, we first compute $\text{Diff}(S, S')$, $\text{Diff}(S, S'')$, $\text{Diff}(R, R')$ and $\text{Diff}(R, R'')$. All these can be computed in polynomial time. Then according to these sets, we identify, through PU1 to PU5, $\text{Diff}_{PU_i}(M, M')$ and $\text{Diff}_{PU_i}(M, M'')$ ($i = 1, \dots, 5$). Again, these can also be done in polynomial time. Finally, by checking $\text{Diff}_{PU_i}(M, M'') \subseteq \text{Diff}_{PU_i}(M, M')$ ($i = 1, \dots, 5$) we can decide whether $M'' <_M M'$. So both steps (1) and (2) can be achieved in polynomial time with a non-deterministic Turing machine.

Hardness proof. It is well known that the validity problem for a propositional formula is co-NP-complete. Given a propositional formula ϕ , we construct in polynomial time a transformation from the problem of deciding ϕ 's validity to a CTL model update. Let X be the set of all variables occurring in ϕ , and a, b two new variables not occurring in X . We denote $\neg X = \bigwedge_{x_i \in X} \neg x_i$. We specify a CTL Kripke model based on the variable set $X \cup \{a, b\}$: $M = (\{s_0, s_1\}, \{(s_0, s_1), (s_1, s_1)\}, L)$, where $L(s_0) = \emptyset$ (i.e. all variables are assigned false), $L(s_1) = X$ (i.e. variables in X are assigned true, while a, b are assigned false). Now we define a new formula $\mu = \text{EX}((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg \phi \wedge a)$. Clearly, formula $((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg \phi \wedge a)$ is satisfiable and $s_1 \not\models ((\phi \supset a) \wedge (\neg X \wedge b)) \vee (\neg \phi \wedge a)$. So $(M, s_0) \not\models \mu$. Consider the update $\text{Update}((M, s_0), \mu)$. We define $M' = (\{s'_0, s'_1\}, \{(s'_0, s'_1), (s'_1, s'_1)\}, L')$, where $L'(s'_0) = L(s_0)$ and $L'(s'_1) = \{a, b\}$. Now we show that ϕ is valid iff (M', s'_0) is an admissible update result from $\text{Update}((M, s_0), \mu)$.

Case 1. We show that if ϕ is valid, then (M', s'_0) is an admissible update result from $\text{Update}((M, s_0), \mu)$. Since ϕ is valid, we have $\neg X \models \phi$. So we have $s'_1 \models (\phi \supset a) \wedge (\neg X \supset b)$. This follows $(M', s'_0) \models \mu$. Also note that M' is obtained by applying PU3 to substitute s_1 with s'_1 . $\text{Diff}(s_1, s'_1) = X \cup \{a, b\}$ which presents a minimal change from s_1 in order to satisfy $(\phi \supset a) \wedge (\neg X \supset b)$.

Case 2. Suppose that ϕ is not valid. Then there exists $X_1 \subseteq X$ such that $X_1 \models \neg \phi$. We construct $M'' = (\{s''_0, s''_1\}, \{(s''_0, s''_1), (s''_1, s''_1)\}, L'')$, where $L''(s''_0) = L(s_0)$ and $L''(s''_1) = X_1 \cup \{a\}$. It is easy to see that $s''_1 \models (\neg \phi \wedge a)$, and hence $(M'', s''_0) \models \mu$. Now we show that $(M', s'_0) \models \mu$ implies $M'' <_M M'$. Suppose $(M', s'_0) \models \mu$. Clearly, both M' and M'' are obtained from M by applying PU3 one time on s_1 respectively. But we have $\text{Diff}(s_1, s'_1) = (X - X_1) \cup \{a\} \subset X \cup \{a, b\} = \text{Diff}(s_1, s''_1)$. This concludes that (M', s'_0) is not an admissible update model. \square

Theorem 4 implies that it is probably not feasible to develop a polynomial time algorithm to implement our CTL model update. Indeed, our algorithm described in the next section, generally runs in exponential time. Nevertheless, we can identify certain typical update cases that may be achieved in polynomial time.

Theorem 5 *Let $M = (S, R, L)$ be a CTL Kripke model and ϕ a CTL formula. If an admissible update result $\text{Update}((M, s_0), \phi)$ ($s_0 \in S$) can be obtained by only applying PU1, PU2, PU4 and PU5 without path selection in M (i.e. ϕ does not involve $\text{EG}\psi$, $\text{EF}\psi$ or $E[\psi_1 \cup \psi_2]$), then this result can be computed in polynomial time.*

6 The Algorithm and a Case Study

A prototype for the CTL model update addressed in previous sections has been implemented recently. The algorithm is designed following a similar style of CTL model checking algorithm SAT [8], where an updated formula is parsed through its structure and recursive calls to proper functions are made to its sub-formulas.

$\text{CTLUpdate}(\mathcal{M}, \phi) / \mathcal{M} = (M, s_0) \not\models \phi$. Update \mathcal{M} to satisfy ϕ . /
Input: $M = (S, R, L)$, $\mathcal{M} = (M, s_0)$, where $s_0 \in S$ and $\mathcal{M} \not\models \phi$
Output: $M' = (S', R', L')$, $\mathcal{M}' = (M', s'_0)$, $s'_0 \in S'$, $\mathcal{M}' \models \phi$;
{ case
 ϕ is \perp : return $\{\mathcal{M}\}$;
 ϕ is atomic p : return $\{\text{Update}_p(\mathcal{M}, p)\}$;
 ϕ is $\neg \phi_1$: return $\{\text{Update}_{\neg}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\phi_1 \vee \phi_2$:
return $\{\text{CTLUpdate}(\mathcal{M}, \phi_1) \text{ or } \text{CTLUpdate}(\mathcal{M}, \phi_2)\}$;
 ϕ is $\phi_1 \wedge \phi_2$: return $\{\text{Update}_{\wedge}(\mathcal{M}, \phi_1, \phi_2)\}$;
 ϕ is $\text{EX}\phi_1$: return $\{\text{Update}_{\text{EX}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\text{AX}\phi_1$: return $\{\text{Update}_{\text{AX}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\text{EF}\phi_1$: return $\{\text{Update}_{\text{EF}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\text{AF}\phi_1$: return $\{\text{Update}_{\text{AF}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\text{EG}\phi_1$: return $\{\text{Update}_{\text{EG}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $\text{AG}\phi_1$: return $\{\text{Update}_{\text{AG}}(\mathcal{M}, \phi_1)\}$;
 ϕ is $E(\phi_1 \cup \phi_2)$: return $\{\text{Update}_{\text{EU}}(\mathcal{M}, \phi_1, \phi_2)\}$;
 ϕ is $A(\phi_1 \cup \phi_2)$: return $\{\text{Update}_{\text{AU}}(\mathcal{M}, \phi_1, \phi_2)\}$;
}

Due to a space limit, here we only outline the main idea of implementing functions Update_{\neg} and $\text{Update}_{\text{AG}}$.

$\text{Update}_{\neg}(\mathcal{M}, \phi) / \mathcal{M} \not\models \phi$. Update \mathcal{M} to satisfy ϕ . /
{ case
 ϕ is $\neg p$: return $\{\text{Update}_{\neg p}(\mathcal{M}, p)\}$;
 ϕ is $\neg(\phi_1 \wedge \phi_2) = \neg \phi_1 \vee \neg \phi_2$:
return $\{\text{Update}_{\neg}(\mathcal{M}, \phi_1)\} \text{ or } \{\text{Update}_{\neg}(\mathcal{M}, \phi_2)\}$;
 ϕ is $\neg \text{EF}(\phi_1) = \text{AG}(\neg \phi_1)$: return $\{\text{Update}_{\text{AG}}(\mathcal{M}, \neg \phi_1)\}$;
}

$\text{Update}_{\text{AG}}(\mathcal{M}, \phi) / \mathcal{M} \not\models \text{AG}\phi$. Update \mathcal{M} to satisfy $\text{AG}\phi$. /
{ if $\mathcal{M}_0 = (M, s_0) \not\models \phi$, then PU3 is applied to s_0
such that $\mathcal{M}' = \text{CTLUpdate}(\mathcal{M}_0, \phi)$;
else select a path $\pi = [s_0, s_1, \dots]$, where $\exists s \in \pi$ such that
 $\mathcal{M}_i = (M, s) \not\models \phi$;
select the earliest state $s_i \in \pi$ such that $(M, s_i) \not\models \phi$;
perform one of the following three operations:
(1) applying PU2 to remove relation (s_{i-1}, s_i) ,
obtain result \mathcal{M}' ;
(2) applying PU5 to remove state s_i and
its associated relations, obtain result \mathcal{M}' ;
(3) applying PU3: $\mathcal{M}' = \text{CTLUpdate}(\mathcal{M}_i, \phi)$;

```

if  $\mathcal{M}' \models AG\phi$ , return  $\mathcal{M}'$ ;
else return  $\{Update_{AG}(\mathcal{M}', \phi)\}$ ;
}

```

Theorem 6 Given a Kripke model $M = (S, R, L)$ and a satisfiable CTL formula ϕ such that $(M, s) \not\models \phi$ for some $s \in S$. Then $CTLUpdate((M, s), \phi)$ will return a resulting Kripke model $M' = (S', R', L')$ such that $(M', s') \models \phi$ for some $s' \in S'$ and it is admissible.

In the rest of this section, we will describe some details of our algorithm implementation by applying the model modification to the well known microwave oven scenario [4]. The microwave oven model is described as in Figure 1.

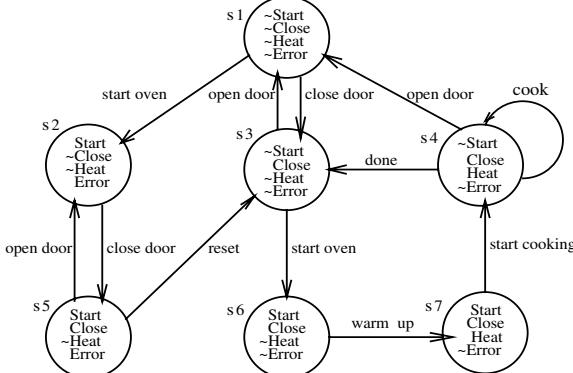


Figure 1. The CTL Kripke structure of a microwave oven.

The Kripke model has 7 states and propositional variables are from the set $\{Start, Close, Heat, Error\}$. It is observed that this model does not satisfy a desirable property $\phi = \neg EF(Start \wedge EG\neg Heat)$ [4]. What we would like to do is to apply our CTL model update program to minimally modify this kripke model to satisfy property ϕ . First, we parse ϕ into $AG(\neg(Start \wedge EG\neg Heat))$ to remove the front \neg . The translation is done by function $Update_{\neg}$. Then we check for each state whether it satisfies $\neg(Start \wedge EG\neg Heat)$. This string is parsed before it is checked. We pick out $EG\neg Heat$ to feed through the model checking function for EG . In this model, any path that has any state with $\neg Heat$ is picked out. Here we find paths $[s_1, s_2, s_5, s_3, s_1, \dots]$ and $[s_1, s_3, s_1, \dots]$ which are strongly connected component loops [4] satisfying $EG\neg Heat$. Then we identify all states with $Start$, they are $\{s_2, s_5, s_6, s_7\}$. Now we select those states with both $Start$ and $\neg Heat$, they are $\{s_2, s_5\}$. Since formula $AG(\neg(Start \wedge EG\neg Heat))$ identifies that the model should not have any states with both $Start$ and $\neg Heat$, we should perform model update related to states s_2 and s_5 .

Now function $Update_{AG}$ starts to perform update as described in the above pseudo code and Theorem 2 (see section 3). It turns out that there are three possible minimal updates: (1) applying PU2 to remove relation (s_1, s_2) ; (2) applying PU5 to remove state s_2 and its associated relations (s_1, s_2) , (s_2, s_5) and (s_5, s_2) ; and (3) applying PU3 on s_2 and s_5 . To perform option (3), our program first converts $\neg(Start \wedge EG\neg Heat)$ to $\neg Start \vee \neg EG\neg Heat$, then s_2 and s_5 are updated with either $\neg Start$ or $\neg EG\neg Heat$ by the main function $CTLUpdate$ to deal with \vee and function $Update_{\neg}$. In our program, updating $\neg Start$ is chosen since formula $\neg Start$ is simpler than $\neg EG\neg Heat$. After the update, we will obtain a resulting model $(M, s_1) \models \phi$. For instance, by choosing update (1) operation above, we will obtain a new Kripke model as shown in Figure 2, which simply states that no state transition from s_1 to s_2 is allowed.

Our algorithm will generate one of the three resulting models without specific indication, because under our criteria they are all minimally changed from the original model. However, in our prototype implementation, we have considered the computational cost of an update upon users' request, which may affect the system efficiency. For instance, in the above example, with three possible updates, if the user requires a low computational cost, our system will select the simplest operation (1) to achieve the goal.

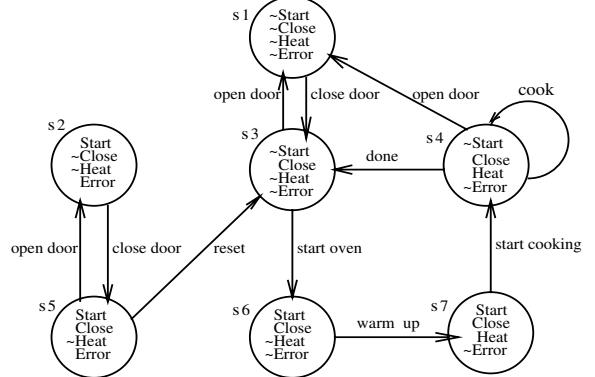


Figure 2. An updated CTL Kripke structure for the microwave oven.

7 Conclusion

In this paper, we presented a formal approach to update CTL models. By specifying five primitive operations on CTL Kripke models, we defined the minimal change criteria for CTL model update. We also investigated the semantic and computational properties of our approach in detail. Based on our formalization, we developed a CTL model update algorithm and implemented a system prototype to perform CTL model updating. Our case study showed an application of this work.

REFERENCES

- [1] C. Baral and Y. Zhang, 'Knowledge updates: semantics and complexity issues', *Artificial Intelligence*, **164**, 209–243, (2005).
- [2] F. Buccafurri, T. Eiter, G. Gottlob, and N. Leone, 'Enhancing model checking in verification by ai techniques', *Artificial Intelligence*, **112**, 57–104, (1999).
- [3] A. Cimatti and et al, 'Nusmv: A new symbolic model verifier', in *Proceedings of the 11th International Conference on Computer Aided Verification*, pp. 495–499, (1999).
- [4] E.Jr. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*, MIT Press, Cambridge, Massachusetts, 2000.
- [5] P. Gammie and R. van der Meyden, 'Mck-model checking the logic of knowledge', in *the Proceedings of the 16th International Conference on Computer Aided Verification*, pp. 479–483, (2004).
- [6] H. Harris and M. Ryan, 'Theoretical foundations of updating systems', in *the Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, pp. 291–298, (2003).
- [7] C. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley Professional, 2003.
- [8] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2000.
- [9] K. McMillan and N. Amla, 'Automatic abstraction without counterexamples', in *Cadence Berkeley Labs, Cadence Design Systems*, (2002).
- [10] J. Wing and M. Vaziri-Farahani, 'A case study in model checking software', in *Proceedings of the 3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering*, (1995).
- [11] M. Winslett, *Updating Logical Databases*, Cambridge University Press, 1990.

Resolving Conflicts in Action Descriptions

Thomas Eiter and Esra Erdem and Michael Fink and Ján Senko¹

Abstract. We study resolving conflicts between an action description and a set of conditions (possibly obtained from observations), in the context of action languages. In this formal framework, the meaning of an action description can be represented by a transition diagram—a directed graph whose nodes correspond to states and whose edges correspond to transitions describing action occurrences. This allows us to characterize conflicts by means of states and transitions of the given action description that violate some given conditions. We introduce a basic method to resolve such conflicts by modifying the action description, and discuss how the user can be supported in obtaining more preferred solutions. For that, we identify helpful questions the user may ask (e.g., which specific parts of the action description cause a conflict with some given condition), and we provide answers to them using properties of action descriptions and transition diagrams. Finally, we discuss the computational complexity of these questions in terms of related decision problems.

1 INTRODUCTION

Action languages [3] are a formal tool for reasoning about actions, where an agent's knowledge about a domain in question is represented by a declarative action description that consists of logical formulas. Consider for instance a light bulb with a switch. When the light is off, then toggling the switch turns the light on; this can be expressed in the action description language \mathcal{C} [4] by the formula

$$\text{caused Light after Toggle} \wedge \neg\text{Light}. \quad (1)$$

On the other hand, at every state, if the light bulb is broken then the light is off. This can be expressed by the formula

$$\text{caused } \neg\text{Light if Broken}. \quad (2)$$

Other pieces of knowledge, like laws of inertia, may be also included. The meaning of such an action description can be represented by a transition diagram—a directed graph whose nodes correspond to the states of the world and the edges to the transitions describing action occurrences. For instance, see Figure 1 for the transition diagram of the action description above (consisting of (1), (2), and inertia laws).

Note that the action description above is “buggy”, since the effects of toggling the switch are not completely specified. Our goal is to “repair” such descriptions taking into account some additional information, such as observations or axioms about the action domain, which can be represented in an action query language [3], expressing conditions on the transition diagram.

For example, when the light bulb is broken, toggling the switch may lead to a state where the light is off; this information is possibly obtained from some observations of the agent, and can be expressed in an action query language, e.g., by the statement

$$\text{possibly } \neg\text{Light after Toggle if Broken}. \quad (3)$$

¹ Institut für Informationssysteme 184/3, Technische Universität Wien, 1040 Wien, Austria, email: {eiter, esra, michael, jan}@kr.tuwien.ac.at

Some of the additional information may conflict with the action description. For instance, condition (3) does not hold relative to the action description above, since at the state where the light bulb is broken and the light is off, toggling the light switch is not possible. Thus, there is a conflict between the action description and this condition.

In this paper, we consider such conflicts, and how the agent's action description can be modified to resolve them. This may be accomplished in many different ways, and there is no canonical method which works satisfactorily in all cases. According to [2], one might aim at dropping a smallest set of candidate formulas to resolve the conflict. In our example, dropping (1) would work. However, under further conditions, like

$$\text{necessarily } \neg\text{Light after Toggle if Light}, \quad (4)$$

the conflict cannot be resolved just by dropping formulas: removing any of (1), (2) and inertia laws will not lead to an edge from a state where the light is on to a state where the light is off. A refined approach is needed which, semantically, modifies the transition diagram by suitable changes of the formulas to “repair” the action description such that the given queries (i.e., conditions) hold.

This paper makes two main contributions in this direction:

- 1) It provides a precise *notion of conflict* between an action description and a set of queries, and presents a basic algorithm to resolve such conflicts. The idea is to modify the transition diagram of the action description by adding or deleting transitions so that all given conditions are satisfied; such a modification of the transition diagram is possible by adding, deleting or modifying some formulas in the action description. Based on this idea, our algorithm calculates a repair whenever it is possible.
- 2) Intuitive repair preferences might be difficult to formalize (e.g., both syntactic and semantic aspects might play a role) and thus to achieve with the basic algorithm above. In such cases, the designer might want to *ask questions* about the action description, the transition diagram, and the extra information, whose answers could guide her to come up with an appealing repair in an iterative repair process. For that, we explore several kinds of such questions and determine properties of action descriptions, transition diagrams, and extra information which are helpful in answering them. We also analyze the computational complexity of related problems.

2 PRELIMINARIES

Transition diagrams. We start with a *propositional action signature* $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$ that consists of a set \mathbf{F} of fluent names, and a set \mathbf{A} of action names. Satisfaction of a propositional formula G over atoms $At \subseteq \mathbf{F} \cup \mathbf{A}$ by an interpretation $P \mapsto I(P) \in \{t, f\}$ for all $P \in At$ as usual, is denoted by $I \models G$. An *action* is an interpretation of \mathbf{A} , denoted by the set of action names that are mapped to t .

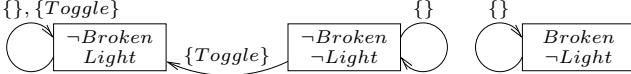


Figure 1. Transition diagram of the action description $\{ (1), (2), (7) \}$.

A transition diagram of \mathcal{L} consists of a set S of states, a function $V : \mathbf{F} \times S \rightarrow \{f, t\}$ such that each state s in S is uniquely identified by the interpretation $P \mapsto V(P, s)$, for all $P \in \mathbf{F}$, and a subset $R \subseteq S \times 2^{\mathbf{A}} \times S$ of transitions. We say that $V(P, s)$ is the value of P in s . The states s' such that $\langle s, A, s' \rangle \in R$ are the possible results of the execution of the action A in the state s . We can think of a transition diagram as a labeled directed graph. Every state s is represented by a vertex labeled with the function $P \mapsto V(P, s)$ from fluent names to truth values; we denote by s the set of fluent literals satisfied by this function. Each triple $\langle s, A, s' \rangle \in R$ is represented by an edge from s to s' and labeled A . See Figure 1 for an example.

Action descriptions. We consider a subset of the action description language \mathcal{C} [4] that consists of two kinds of expressions (called *causal laws*): *static laws* of the form

$$\text{caused } L \text{ if } G, \quad (5)$$

where L is a fluent literal or *False*, and G is a propositional combination of fluent names; and *dynamic laws* of the form

$$\text{caused } L \text{ if } G \text{ after } U, \quad (6)$$

where L and G are as above, and U is a propositional combination of fluent names and action names. In (5) and (6) the part **if** G can be dropped if G is *True*.² An *action description* is a set of causal laws. For instance, one formalization of the light domain described in the introduction can be expressed in this language by the causal laws (1), (2), and the inertia laws

$$\text{inertial } \text{Light}, \neg\text{Light}, \text{Broken}, \neg\text{Broken}. \quad (7)$$

Here an expression of the form **inertial** L_1, \dots, L_k stands for the causal laws **caused** L_i **if** L_i **after** L_i for $i \in \{1, \dots, k\}$.

The meaning of an action description can be represented by a transition diagram as follows. We say that a causal law l is *applicable* to a transition $\langle s, A, s' \rangle$ in a transition diagram, if

- l is a static law (5), such that $s' \models G$; or
- l is a dynamic law (6), such that $s' \models G$ and $s \cup A \models U$.³

We denote by $D(\text{tr})$ the set of all causal laws in an action description D that are applicable to a transition tr ; by $H_D(\text{tr})$ the set of the heads of all causal laws in $D(\text{tr})$; and by $\text{sat}(H_D(\text{tr}))$ the set of interpretations of \mathbf{F} that satisfy $H_D(\text{tr})$.

Let D be an action description with a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Then the transition diagram $\langle S, V, R \rangle$ described by D , denoted $T(D)$, is defined as follows:

- S is the set of all interpretations s of \mathbf{F} such that, for every static law (5) in D , $s \models G \supseteq L$,
- $V(P, s) = s(P)$,
- R is the set of all $\langle s, A, s' \rangle$ such that $\text{sat}(H_D(\langle s, A, s' \rangle)) = \{s'\}$.

We denote by $S(D)$ (resp. $R(D)$) the set of states (resp. transitions) of $T(D)$. For instance the transition diagram described by the action description consisting of (1), (2), (7) is shown in Figure 1.

² *True* (resp. *False*) is the empty conjunction (resp. disjunction).

³ We identify states s with the interpretations $P \mapsto V(P, s)$.

Conditions. For expressing extra conditions, we consider here a language with two kinds of statements (“queries”) about an action description: *possibility* and *necessity queries* of the respective forms

$$\text{possibly } \psi \text{ after } A \text{ if } \phi \quad (8)$$

$$\text{necessarily } \psi \text{ after } A \text{ if } \phi \quad (9)$$

where ϕ and ψ are propositional combinations of fluent names, and A is an action. These queries are syntactically different from the ones presented in [3] and [2]; on the other hand, semantically they constitute a fragment of an extension of the action query language \mathcal{P} [3] (from which we draw the term “query”) and the condition language in [2] (see Related and Further Work for a discussion).

A query q of form (8) (resp., (9)) is *satisfied at state s in a transition diagram T* , denoted $T, s \models q$, if either $s \not\models \phi$, or for some (resp., every) transition $\langle s, A, s' \rangle$ of T $s' \models \psi$ holds. We say that T *entails* a set Q of queries (denoted $T \models Q$), if $T, s \models q$ for every $q \in Q$ and for every state s in T . Accordingly, an action description D *entails* Q (denoted $D \models Q$) if $T(D) \models Q$.

Example 1 Let us consider the light domain described in the introduction as our running example. Let D be the action description consisting of (2), (1), and (7); and Q be the set of two queries: possibility query (3) and the necessity query (4), denoted by q_p and q_n respectively. Figure 1 shows $T(D)$ (i.e., the transition diagram of D). Then it can be easily verified that, at state $\{\neg\text{Light}, \text{Broken}\}$, since there is no transition from this state with action *Toggle*, the query q_n is trivially satisfied while q_p is not satisfied.

What a query describes is different from what a causal law does: action descriptions allow us to describe a transition diagram, based on causal explanations (what “is caused”), whereas queries allow us to state assertions (what “holds”) about transition diagrams. These assertions may, e.g., be observations or axioms about the action domain. (See [3] for a discussion on action query languages.)

3 CONFLICTS IN ACTION DESCRIPTIONS

Given an action description D and a set Q of queries, we say that there is a *conflict* between D and Q , if $D \not\models Q$. Our goal is to resolve these conflicts by modifying the action description.

Conflicts can be characterized, from a semantic point of view, in terms of states and transitions “violating” some queries. We assume that the states of the world are correctly described by the given action description. Thus conflicts are existing transitions (for the violation of a necessity query) and non-existing transitions (for the violation of a possibility query) that cause such conflicts. The idea is then to “repair” an action description by a syntactic modification, such as adding, deleting, or modifying some of its causal laws, so that the detected conflicts are resolved by adding and/or deleting some transitions in the transition diagram.

For an action description D and a set Q of queries, the states and transitions violating possibility and necessity queries in Q , respectively, are as follows.

- A state s of $T(D)$ violates a possibility query q of form (8) in Q , if $T(D), s \not\models q$.
- A transition $\text{tr} = \langle s, A, s' \rangle$ of $T(D)$ violates a necessity query q of form (9) in Q (denoted $\text{tr} \not\models q$), if $s \models \phi$ and $s' \not\models \psi$.

Example 2 (cont’d) From $T(D)$ we can identify the following conflicts: the single state violating the possibility query q_p is $\{\neg\text{Light}, \text{Broken}\}$, and the single transition violating the necessity query q_n is $\{\text{Light}, \neg\text{Broken}\}, \{\text{Toggle}\}, \{\text{Light}, \neg\text{Broken}\}$.

Since we suppose that states of the world are correctly described by D , we do not need to modify the static laws in D for a repair.

Algorithm RESOLVE(D, Q) : $Mod, Incon$ **Input:** An action description, D , and a set of queries, Q .**Output:** A repair, Mod , and a set of queries, $Incon$.

```

 $Mod := \emptyset; Incon := \emptyset;$ 
for all  $(q, tr) \in conf_n(D, Q)$  do
     $Mod := Mod \cup Delete(tr);$ 
 $D' := Mod(D); Ins := \emptyset;$ 
for all  $(q, s) \in conf_p(D', Q)$  do
     $(q = \text{possibly } \psi \text{ after } A \text{ if } \phi)$ 
     $Cands := \{\langle s, A, s' \rangle \mid s' \in S(D), s' \models \psi,$ 
     $\langle s, A, s' \rangle \models q', \forall q' \in Q_n\};$ 
    if ( $Cands \neq \emptyset$ ) then
        select  $tr \in Cands;$ 
         $Ins := Ins \cup \{tr\};$ 
    else
         $Incon := Incon \cup \{q\};$ 
return  $Mod \cup Insert(Ins, D'), Incon;$ 

```

Figure 2. An algorithm to resolve conflicts.

4 A METHOD FOR RESOLVING CONFLICTS

Under the assumption above, we can resolve conflicts between an action description D and a set Q of queries by the algorithm presented in Figure 2. Before we explain how this algorithm works, let us describe the notation used in it.

For a set Q of queries, we denote by Q_p (resp. Q_n) the set of possibility (resp. necessity) queries in Q . Then

$$\begin{aligned} conf_p(D, Q) &= \{(q, s) \mid q \in Q_p, s \in S(D), T(D), s \not\models q\} \\ conf_n(D, Q) &= \{(q, tr) \mid q \in Q_n, tr \in R(D), tr \not\models q\}. \end{aligned}$$

For a triple $tr = \langle s, A, s' \rangle$, where s and s' are states and A is an action, and a dynamic causal law $l = \text{caused } L \text{ if } U \text{ after } G$, $\langle s, A, s' \rangle \models l$ if either l is not applicable to tr , or $s' \models l$.

A *repair item* is an expression of form $(modify, l, l')$, or (add, l) , where l and l' are dynamic causal laws. A *repair* is a set of repair items. For an action description D and a repair M , we denote by $M(D)$ the action description obtained from D by applying the modifications specified by the repair items in a repair M : (add, l) modifies D by adding l ; $(modify, l, l')$ modifies D by replacing l with l' ; all repair items are executed in parallel, i.e., if M comprises several *modify* items for the same law l , all corresponding modifications l' are generated and eventually replace l . The repairs used by the algorithm RESOLVE(D, Q) are as follows (in causal laws, a state s stands for $\bigwedge_{L \in s} L$, and an action A for $\bigwedge_{X \in A} X \wedge \bigwedge_{X \in A \setminus A} \neg X$):

$$\begin{aligned} Delete(\langle s, A, s' \rangle) &= \{(add, \text{caused } False \text{ if } s' \text{ after } A \wedge s)\} \\ Insert(Tr, D) &= \{(add, \text{caused } L \text{ if } s' \text{ after } A \wedge s) \mid \langle s, A, s' \rangle \in Tr, L \in s'\} \cup \\ &\quad \{(modify, l, \text{caused } L \text{ if } G \text{ after } U \wedge \alpha(Tr, l)), \\ &\quad (modify, l, \text{caused } L \text{ if } G \wedge L \text{ after } U \wedge A \wedge s) \mid \\ &\quad l = \text{caused } L \text{ if } G \text{ after } U, l \in D, \langle s, A, s' \rangle \in Tr, \langle s, A, s' \rangle \not\models l\} \end{aligned}$$

where $\alpha(Tr, l) = \bigwedge_{\langle s, A, s' \rangle \in Tr, \langle s, A, s' \rangle \not\models l} \neg A \vee \neg s$.

In the algorithm above, first every transition tr violating the necessity queries in Q is removed, by adding to D the causal laws $Delete(tr)$. The new action description, D' , entails Q_n . Then, for each state s violating a possibility query $q = \text{possibly } \psi \text{ after } A \text{ if } \phi$ in Q relative to D' , a set $Cands$ of transition candidates tr (triples of form $\langle s, A, s' \rangle$ where $s' \in S(D)$) that, when added to $T(D')$, would

satisfy q at s (i.e., $s' \models \psi$) but not violate any necessity queries in Q (i.e., $tr \models q', \forall q' \in Q_n$), is computed. If such transition candidates exist (i.e., $Cands \neq \emptyset$), by introducing only one of these candidates into $T(D')$, the violation of q at s is prevented; otherwise no repair of D exists for Q (i.e., $Incon$ is not empty, and it contains the possibility queries that conflict with some necessity query in Q). The set Ins denotes all the transition candidates to be introduced into $T(D')$ so that no possibility query is violated in any state. Adding Ins to $T(D')$ can be achieved by adding to D' the causal laws $Insert(Ins, D')$.

Theorem 1 For any repair Mod and set $Incon$ of queries output by RESOLVE(D, Q), the following hold:

1. $D \models Q$ iff $Mod = \emptyset$ and $Incon = \emptyset$;
2. $Incon = \emptyset$ iff $\exists D'$ such that $S(D) = S(D')$ and $D' \models Q$;
3. if $Incon = \emptyset$, then $Mod(D) \models Q$.

The selection of a transition candidate $tr \in Cands$ for repairing a possibility query constitutes a choice point of the algorithm, where further heuristics can be employed to prune the set of repairs. We could, e.g., prefer transition candidates that *respect inertia conditions* or compute *minimal modifications*, i.e., repairs such that the modifications to $T(D)$ are minimal w.r.t. addition or deletion of transitions.

Example 3 (cont'd) Stipulating preference of transition candidates that respect inertia, the basic method resolves the conflicts as follows. First, the only transition violating q_n (i.e., $\langle s_1, \{ \text{Toggle} \}, s_1 \rangle$, where $s_1 = \{ \text{Light}, \neg \text{Broken} \}$) is deleted from $T(D)$ by adding the law:

caused False if Light $\wedge \neg \text{Broken}$ after Toggle $\wedge \text{Light} \wedge \neg \text{Broken}$.

Then, to resolve conflicts with q_p , the only transition candidate respecting inertia (i.e., $tc = \langle s_2, \{ \text{Toggle} \}, s_2 \rangle$, where $s_2 = \{ \neg \text{Light}, \text{Broken} \}$) is introduced into $T(D')$ ($Insert(tc, D') = \{(add, l_i, (modify, (1), l_j)) \mid i = 1, 2, j = 3, 4\}$), replacing (1) with the laws:

$$\begin{aligned} l_1 &: \text{caused } \neg \text{Light} \text{ if } \neg \text{Light} \wedge \text{Broken} \text{ after } \\ &\quad \text{Toggle} \wedge \neg \text{Light} \wedge \text{Broken}, \\ l_2 &: \text{caused } \text{Broken} \text{ if } \neg \text{Light} \wedge \text{Broken} \text{ after } \\ &\quad \text{Toggle} \wedge \neg \text{Light} \wedge \text{Broken}, \\ l_3 &: \text{caused } \text{Light} \text{ if } \text{Light} \text{ after } \text{Toggle} \wedge \neg \text{Light} \wedge \text{Broken}, \\ l_4 &: \text{caused } \text{Light} \text{ after } \text{Toggle} \wedge \neg \text{Light} \wedge \neg \text{Broken}. \end{aligned}$$

We remark that algorithm RESOLVE can be implemented to use polynomial work space, producing its output, which is exponential in general, as a stream. After computing RESOLVE(D, Q), to get a more concise description, one may drop redundant causal laws that might have been introduced (e.g., (6) where $U \equiv \text{False}$), and apply some equivalence preserving transformations (e.g., replacing two laws **caused L after $A \wedge U$** and **caused L after $A \wedge \neg U$** with **caused L after A** .) Note also, that if there exists a repair for D , then there always also exists a repair D' of polynomial size. Informally speaking, D' can be obtained by expressing all necessity queries as dynamic laws and dispensing causality for all actions occurring in queries (**caused L if L after A** , for every literal L). Such a repair is independent of D apart from static laws and semantically it amounts to a complete transition graph w.r.t. actions occurring in queries modulo transitions violating necessity queries. Thus, it is even less appealing than solutions computed by RESOLVE(D, Q), which aim at making modifications as local as possible on single transitions (in order to retain the original semantics of D as much as possible even in case of further modifications). In most cases, however, neither of these basic repairs will be satisfactory. This motivates the utilization of additional knowledge of certain properties for repair.

5 TOWARDS USER-ASSISTED REPAIRS

With the method described above we can automatically repair an action description D with respect to a set Q of queries, under the assumption that the states of the world are described correctly by D . However, we may end up with an action description with many causal laws, some possibly redundant or implausible. To get a more appealing description most often requires respecting additional knowledge or intuitions of the *designer* about the action description.

Usually, this knowledge cannot be easily formalized, as the following example illustrates:

Example 4 The designer of D might use her knowledge about the domain, i.e., light bulbs and switches, to infer from the conflict with the observation expressed in q_n that the duality of the toggle action has not been modeled correctly, and that the conflict with q_p is due to neglecting the effects of toggling when the bulb is broken. Hence, instead of D , she might consider D' consisting of (2), (7), and:

$$\begin{aligned} \text{caused } & \text{Light after Toggle} \wedge \neg \text{Light} \wedge \neg \text{Broken} \\ \text{caused } & \neg \text{Light after Toggle} \wedge \text{Light} \wedge \neg \text{Broken}. \end{aligned} \quad (10)$$

Note that this description is more concise and plausible than the one generated by the basic method (see Example 3).

For (interactively) providing support to a designer repairing an action description, we present some questions that she may ask about Q , D , and $T(D)$. Answers to these questions are obtained from useful properties of queries, action descriptions, and transition diagrams.

Questions about queries and causal laws. To better understand the reasons for conflicts, the designer may want to check the given queries Q make sense with each other. Then the question is:

D1: If Q is contradictory relative to D , which queries in Q are contradictory?

We understand contradiction in a set Q as follows:

Definition 1 A set Q of queries is contradictory relative to an action description D , if there is no action description D' such that $S(D) = S(D')$ and $D' \models Q$.

With an answer to **D1**, the designer may drop contradictory queries from Q . Here are some sufficient conditions to find these queries.

Proposition 1 A set Q of queries is contradictory relative to D , if Q includes a possibility query of form (8) such that some $s \in S(D)$ satisfies ϕ , but no $s \in S(D)$ satisfies ψ .

Proposition 2 A set Q of queries is contradictory relative to D , if Q includes a necessity query necessarily ψ' after A if ϕ' and a possibility query (8) such that some state in $S(D)$ satisfies $\phi \wedge \phi'$, but no state in $S(D)$ satisfies $\psi \wedge \psi'$.

Example 5 In our running example (i.e., Example 1), if Q had contained the query possibly $\text{Light} \wedge \text{Broken}$ after Toggle if True then, due to Proposition 1, Q would be contradictory relative to D .

If the given set of queries is not contradictory, then she may ask:

D2: If D does not satisfy a particular necessity query q in Q , which dynamic causal laws in D violate q ?

We understand violation of a query as follows:

Definition 2 A dynamic causal law $l \in D$ violates a given necessity query q , if there is a transition $tc = \langle s, A, s' \rangle$ in $T(D)$ such that tc violates q , l is applicable to tc , and s' satisfies the head of l .

Once the designer finds out which causal laws violate which queries, she may want to repair the action description in a way that some causal laws (e.g., the inertia laws) are not modified at all:

D3: Can we resolve a conflict between D and Q , without modifying a set D_0 of causal laws in D ?

To answer **D3** the following definition and proposition are helpful.

Definition 3 A transition diagram T satisfies a set D of causal laws (denoted $T \models D$), if, for each transition $tc = \langle s, A, s' \rangle$ in T , for each causal law $l \in D$, l is not applicable to tc or s' satisfies the head of l .

Proposition 3 Let D be an action description, and Q be a set of queries. If there exists a transition diagram T such that $T \models D$ and $T \models Q$, then there exists an action description D' , such that $S(D) = S(D')$, $D \subseteq D'$ and $T = T(D')$.

With this proposition, we can answer **D3** by checking if any transition diagram, having states $S(D)$, that satisfies D_0 also entails Q .

Example 6 In our running example it is possible to repair D without modifying the inertia laws: there exists an action description containing the inertia laws and satisfying the given queries (cf. Example 4).

In another scenario, the designer may suspect that the definition of a particular fluent causes problems, so she may want to know whether some particular laws have to be modified in order to obtain a repair:

D4: Do we have to modify a set D_0 of dynamic causal laws in D to resolve a conflict between D and Q ?

For this, due to the proposition below, we can check whether none of the transition diagrams, with the same states as D (and thus as D_0), that satisfy D_0 , entails Q .

Proposition 4 Let D_0 be an action description, and Q be a set of queries. If some $D_0 \subseteq D$ satisfies $S(D_0) = S(D)$ and $D \models Q$, then there exists a transition diagram T such that $T \models D_0$ and $T \models Q$.

Questions about states and transitions. Alternatively, the designer may want to extract some information from $T(D)$. For instance, an answer to the following question gives information about states violating a query q in Q :

T1: Which states of $T(D)$ satisfying a given formula ϕ' violate q ?

Example 7 In Example 1, if we just consider states where the light is on (i.e., $\phi' = \text{Light}$), then the only state at which a query of Q is violated is $\{\text{Light}, \neg \text{Broken}\}$.

An answer to the following question gives information about transitions violating a necessity query q in Q :

T2: Given formulas ψ' and ϕ' , which transitions $\langle s, A, s' \rangle$ of $T(D)$ such that s satisfies ϕ' and s' satisfies ψ' , violate q ?

With such information extracted from the transition diagram, the designer might decide how to modify the action description D .

Suppose that D does not satisfy a possibility query (8) in Q . The designer may want to learn about possible transition candidates that, when added to $T(D)$ by modifying the definition of some literal L in D , might lead to a repair:

T3: Given a literal L , for every state s of $T(D)$ such that s satisfies ϕ , is there some under-specified transition candidate $tc = \langle s, A, s' \rangle$ for D such that s' satisfies $\psi \wedge L$ and L is under-specified relative to tc ? If there is, then what are they?

Here under-specification is understood as follows:

Definition 4 A transition candidate $tc = \langle s, A, s' \rangle$ for D is under-specified, if $\{s'\} \subset \text{sat}(H_D(tc))$. A literal L is under-specified relative to a transition candidate tc , if $\{L, \bar{L}\} \cap H_D(tc) = \emptyset$.

With a positive answer to **T3**, the designer may try to modify the description D , e.g., by adding caused L if ψ after $A \wedge \phi$.

6 COMPLEXITY RESULTS

First let us remind the following result from [2]: Given an action description D and a set Q of queries, deciding $D \models Q$ is Π_2^p -complete in general. When Q contains the single query **possibly True after A if True**, the executability of an action A at every state, this result conforms with the ones reported in [10, 6].

In the following, we formally state two central results and, informally discuss how to obtain further results. The first main result is about the existence of a conflict resolution between an action description D and Q without modifying a subset D_0 of D .

Theorem 2 *Given D, Q , and $D_0 \subseteq D$, deciding if there exists some D' , such that $S(D)=S(D')$, $D_0 \subseteq D'$, and $D' \models Q$, is Π_2^p -complete.*

We can show Π_2^p -hardness even for $D_0 = \emptyset$; for such D_0 , complexity drops only if in addition Q is restricted to queries of form (8) (to P_{\parallel}^{NP} -completeness, i.e., polynomial time with parallel queries to an NP-oracle, see, e.g., [5]).

The second main result is about the existence of a conflict resolution between an action description D and Q without modifying the transition diagram described by D .

Theorem 3 *Given D and Q , deciding if there exists some D' , such that $S(D)=S(D')$, $D' \models Q$, and $R(D) \subseteq R(D')$, is Π_2^p -complete.*

We remark that if some repair of D for Q is known to exist, then deciding the above problem is coNP-complete.

Table 1. Complexity results (completeness) for problems **D1–D4**, **T1–T3**.

Problem	D1	D2	D3	D4	T1	T2	T3
	Σ_2^p	NP	Π_2^p	Σ_2^p	Σ_2^p	NP	Π_2^p
$Q_n = \emptyset$	P_{\parallel}^{NP}	$O(1)$	Π_2^p	Σ_2^p	Σ_2^p	$O(1)$	Π_2^p
$Q_p = \emptyset$	$O(1)$	NP	$O(1)$	$O(1)$	NP	NP	Π_2^p

Table 1 shows complexity results for the decision problems resp. existence problems⁴ related to the questions above (denoted **D1–D4**, resp. **T1–T3**) for the general case, and when $Q_n = \emptyset$, or $Q_p = \emptyset$.

Deciding whether Q is contradictory w.r.t. D (**D1**) is Σ_2^p -complete in general. Intuitively, this is because deciding the violation of a possibility query q is Σ_2^p -complete. We have to guess a violating state and verify, by means of an NP-oracle, for corresponding transition candidates that they do not satisfy q . Since we can express necessity queries by dynamic causal laws, this source of complexity carries over to deciding whether a set of (mixed) queries Q is contradictory. From these observations, Σ_2^p -completeness of the existence version of **T1** (i.e., whether such a state exists) is straightforward. However, if $Q_n = \emptyset$, to show that Q is contradictory w.r.t. D , it is sufficient to test whether, for some query (8) in Q_p , some state satisfies ϕ but no state satisfies ψ . This amounts to a Boolean combination of SAT instances, whose evaluation is in P_{\parallel}^{NP} . For $Q_p = \emptyset$, note that a set Q_n of necessity queries cannot be contradictory.

On the other hand, deciding whether a necessity query q is violated is in NP: Guess and verify in polynomial time a transition violating q . Thus, e.g., deciding whether a causal law $l \in D$ violates $q \in Q_n$ (i.e., **D2**) is NP-complete, as well as the existence version of **T2**.

D3 is the problem considered in Theorem 2, **D4** is the complementary problem, and corresponding results have been discussed above. Finally, the property of **T3** fails if there exists a state s satisfying ϕ , such that no under-specified transition candidate $tc = \langle s, A, s' \rangle$ for D exists, such that $s' \models \psi \wedge L$. Since for a given s , this can be checked with an NP-oracle, failure of the property is in Σ_2^p .

⁴ Formal statements of these problems will be given in an extended version.

7 RELATED AND FUTURE WORK

In [2], the authors describe a method to minimally modify an action description, when new causal laws are added, by deleting some causal laws, so that given queries are satisfied. In the method above, we obtain an action description by adding or modifying some causal laws, motivated by some reasons for conflicts. For some problems, as discussed in the introduction, just dropping causal laws as in [2] does not lead to a solution, whereas our method above does.

Similar to [2], [8] discusses how to minimally update a logic program syntactically so that given observations are satisfied. A semantical approach to updating a logic program by changes to Kripke structures is given in [9], but no conditions are considered. [11] describes how to resolve conflicts between a logic program and a set of constraints by “forgetting” some atoms in the program; [12] describes how logic programs can be updated with that approach.

In [1], the authors extend an action description, encoded as a logic program, with “consistency restoring” rules, so that when the action description and given observations are incompatible, these rules can be “applied” to get some consistent answer set. This, however, is more geared towards handling exceptions. Lifschitz describes in [7] an action domain in language C such that every causal law is defeasible (by means of an abnormality predicate). To formulate some other variations of the domain, the agent can just add new causal laws, some of which “disable” some existing causal laws. In [1] and [7], the causal laws of the original domain description are not modified.

Ongoing work includes an implementation of the method described above for resolving conflicts, and the investigation of the use of a SAT solver or an answer set solver to answer the questions discussed above (as suggested by the computational complexity results of the corresponding decision problems, presented in Table 1). Employing a richer query language, like that of [2] or the extension of action query language \mathcal{P} as in [3], in which conditions on sequences of action occurrences can be expressed, is a future work.

ACKNOWLEDGEMENTS

We thank the anonymous referees for their comments and suggestions. This work is supported by FWF grant P16536-N04.

REFERENCES

- [1] M. Balduccini and M. Gelfond, ‘Logic programs with consistency-restoring rules’, in *Work. notes of AAAI Spring Symp.*, pp. 9–18, (2003).
- [2] T. Eiter, E. Erdem, M. Fink, and J. Senko, ‘Updating action domain descriptions’, in *Proc. IJCAI*, pp. 418–423, (2005).
- [3] M. Gelfond and V. Lifschitz, ‘Action languages’, *ETAI*, 3, 195–210, (1998).
- [4] E. Giunchiglia and V. Lifschitz, ‘An action language based on causal explanation: Preliminary report’, in *Proc. AAAI*, pp. 623–630, (1998).
- [5] D. S. Johnson, ‘A catalog of complexity classes’, in *Handbook of Theoretical Computer Science*, vol. A, 67–161, MIT Press, (1990).
- [6] J. Lang, F. Lin, and P. Marquis, ‘Causal theories of action: A computational core’, in *Proc. IJCAI*, pp. 1073–1078, (2003).
- [7] V. Lifschitz, ‘Missionaries and cannibals in the causal calculator’, in *Proc. KR*, pp. 85–96, (2000).
- [8] C. Sakama and K. Inoue, ‘An abductive framework for computing knowledge base updates’, *TPLP*, 3(6), 671–713, (2003).
- [9] J. Sefránek, ‘A Kripkean semantics for dynamic logic programming’, in *Proc. LPAR*, pp. 469–486, (2000).
- [10] H. Turner, ‘Polynomial-length planning spans the polynomial hierarchy’, in *Proc. JELIA*, pp. 111–124, (2002).
- [11] Y. Zhang, N. Foo, and K. Wang, ‘Solving logic program conflicts through strong and weak forgettings’, in *Proc. IJCAI*, pp. 627–632, (2005).
- [12] Y. Zhang and N. Foo, ‘A unified framework for representing logic program updates’, in *Proc. AAAI*, pp. 707–713, (2005).

Possibilistic Influence Diagrams

Laurent GARCIA¹ and Régis SABBADIN²

Abstract. In this article we present the framework of *Possibilistic Influence Diagrams* (PID), which allow to model in a compact form problems of sequential decision making under uncertainty, when only ordinal data on transitions likelihood or preferences are available. The graphical part of a PID is exactly the same as that of usual influence diagrams, however the semantics differ. Transition likelihoods are expressed as possibility distributions and rewards are here considered as satisfaction degrees. Expected utility is then replaced by anyone of two possibilistic qualitative utility criteria for evaluating strategies in a PID. We describe a decision tree-based method for evaluating PID and computing optimal strategies. We then study the computational complexity of PID-related problems (computation of the value of a policy, computation of an optimal policy).

Keywords: decision, possibility theory, causal networks, influence diagrams.

1 INTRODUCTION

For several years, there has been a growing interest in the Artificial Intelligence community towards the foundations and computational methods of decision making under uncertainty. This is especially relevant for applications to planning, where a suitable sequence of decisions is to be found, starting from a description of the initial world, of the available decisions and their effects, and of the goals to reach. Several authors have thus proposed to integrate some parts of decision theory into the planning paradigm. A classical model for decision making under uncertainty is based on *Markov decision processes* (MDP), where actions effects are stochastic and the satisfaction of agents expressed by a numerical, additive utility function. However, classical dynamic programming (DP) algorithms [9] do not scale to the size of problems that can be compactly represented by the factored representations traditionally used in AI. Therefore several methods of aggregation/decomposition for large problems have been proposed in the AI community. Aggregation methods [2], for example, use a compact representation of probability and reward functions involved in the MDP description and propose algorithms dealing explicitly with this representation. These methods are often based on the use of *Influence Diagrams* [6] which form such a factored representation framework for problems of decision under uncertainty (sequential or non sequential).

However, transition probabilities are not always available for representing the effects of actions, especially in Artificial

Intelligence applications where uncertainty is often ordinal, qualitative. The same remark applies to utilities: it is often more adequate to represent preference over states simply with an ordering relation rather than with additive utilities. Several authors have advocated this qualitative view of decision making and have proposed qualitative versions of decision theory. [4] propose a qualitative utility theory based on possibility theory, where preferences and uncertainty are both expressed qualitatively. [5, 10] have extended the use of these criteria to sequential decision, allowing the definition of (non-factored) possibilistic MDP and have proposed counterparts of the classical DP solution methods for MDP.

In this paper, after having presented possibilistic counterparts of expected utility (Section 2), we present (Section 3) the framework of *Possibilistic Influence Diagrams* (PID), which allow to model in a compact form problems of sequential decision making under uncertainty, where only ordinal data on transition likelihoods or preferences are available. The graphical part of PID is exactly the same as that of usual influence diagrams, however the semantics differ. Transition likelihoods are expressed by possibility distributions, and rewards are here considered as satisfaction degrees attached to partial goals. Expected utility is then replaced by anyone of the two possibilistic qualitative utility criteria proposed by [4] for evaluating strategies in a PID. Then (Section 4), we describe a decision tree-based method for evaluating PID and computing optimal strategies. We finally show some complexity results on solving PID-related problems (Section 5).

2 POSSIBILISTIC COUNTERPARTS OF EXPECTED UTILITY

[4] propose an ordinal counterpart of the expected utility theory based on possibility theory. The basic measure of possibility theory is the possibility distribution which describes knowledge about the unknown value taken by one or several attributes used to describe states of affairs. It can represent *uncertain knowledge* distinguishing what is plausible from what is less or it can represent *preferences* over desired or less desired states.

In the framework of decision under uncertainty, some state variables are *controlled* by a decision maker who can choose their value, depending on the observation of the values of (all or) some other state variables. The decision maker thus applies a *strategy* δ mapping the set of (observed) uncontrolled state variables values into controlled (decision) variables values. The uncertainty of the agent about the effect of strategy δ is represented by a possibility distribution π_δ mapping the set X of state variables values into a bounded, linearly ordered

¹ LERIA, University of Angers, France. Email: laurent.garcia@info.univ-angers.fr

² INRA-BIA Toulouse, France. Email: sabbadin@toulouse.inra.fr

(qualitative) valuation set L . Set L is assumed to be equipped with an order-reversing map n . Let 1_L and 0_L denote the top and bottom elements of L , respectively. Then $n(0_L) = 1_L$ and $n(1_L) = 0_L$. The quantity $\pi_\delta(x)$ thus represents the degree of possibility of the state x being the consequence of strategy δ . $\pi_\delta(x) = 1_L$ means that x is completely plausible, whereas $\pi_\delta(x) = 0_L$ means that it is completely impossible.

It makes sense, if information is qualitative, to represent not only the incomplete knowledge on the state by a possibility distribution π_δ on X but also the decision-maker's preferences on X by means of another possibility distribution μ_δ with values on a preference scale U . Here, we assume that uncertainty and preferences are commensurate, that is U can be identified to L . $\mu_\delta(x) = 1_L$ means that x is completely satisfactory, whereas if $\mu_\delta(x) = 0_L$, it is totally unsatisfactory. Notice that π_δ is normalised (there shall be at least one completely possible state of the world), but μ_δ may not be (it can be that no consequence is totally satisfactory).

[4] proposed the two following qualitative decision criteria for assessing the value of a strategy δ :

$$\begin{aligned} u^*(\delta) &= \max_{x \in X} \min\{\pi_\delta(x), \mu_\delta(x)\} \\ u_*(\delta) &= \min_{x \in X} \max\{n(\pi_\delta(x)), \mu_\delta(x)\} \end{aligned}$$

u^* can be seen as an extension of the *maximax* criterion which assigns to an action the utility of its best possible consequence. On the other hand, u_* is an extension of the *maximin* criterion which corresponds to the utility of the worst possible consequence. u_* measures to what extent every plausible consequence is satisfactory. u^* corresponds to an adventurous (optimistic) attitude in front of uncertainty, whereas u_* is conservative (cautious).

3 POSSIBILISTIC INFLUENCE DIAGRAMS

Possibilistic influence diagrams are the possibilistic counterpart of *influence diagrams* (ID) [6, 11] which extend Bayesian networks to handle decision under uncertainty. A *possibilistic influence diagram* (PID, see Figure 1) is a *directed acyclic graph* DAG containing three different kinds of nodes :

- **chance nodes**, drawn as circles, represent state variables $X_i \in \mathcal{X} = \{X_1, \dots, X_n\}$, as in the *Bayesian Networks* (BN) framework [8]. A combination $x = \{x_1, \dots, x_n\}$ of state variables values represents a state.
- **decision nodes**, drawn as rectangles, represent decision variables $D_i \in \mathcal{D} = \{D_1, \dots, D_p\}$. A combination $d = \{d_1, \dots, d_p\}$ of values represents a decision.
- **utility nodes** $\{V_1, \dots, V_q\}$, drawn as diamonds, represent local “satisfaction degree” functions $r_i \in \{r_1, \dots, r_q\}$.

The directed edges in the PID represent either causal influences or information influences between variables.

Before we describe the underlying semantics of possibilistic influence diagrams (PID), let us give two structural assumptions on the DAG : i) The utility nodes have no children. ii) There exists a directed path comprising all decision nodes³.

³ While the first assumption is classical in usual ID, the second one is simply a convention which simplifies the resolution of problems described as ID (possibilistic or not). The directed path induces an ordering of decision variables which will be exploited by the decision tree solution methods.

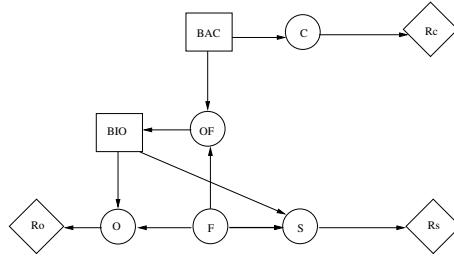


Figure 1. How to make a six-egg omelette from a five-egg one

Now, in addition to the graphical part, which is very similar to that of a classical ID, PID also comprise numerical (ordinal) specifications. Let us recall that, in our possibilistic setting, this numerical ordering is handled in a qualitative way. First, state and decision variables have a finite state space. Furthermore, utility nodes have no state space attached. Then, concerning local dependencies, each state variable X_i has a set $Par(X_i)$ of parents in the DAG. $Par(X_i)$ can comprise both state and decision variables. A conditional possibility table $\Pi(X_i|Par(X_i))$ is attached to every state variable X_i . It specifies every conditional possibility $\Pi(x_i|x_{Par(X_i)}, d_{Par(X_i)})$ where $x_i \in X_i$ ⁴, $x_{Par(X_i)} \in \mathcal{X}_{Par(X_i)} = \bigotimes_{j, X_j \in Par(X_i) \cap \mathcal{X}} X_j$ and $d_{Par(X_i)} \in \mathcal{D}_{Par(X_i)} = \bigotimes_{j, D_j \in Par(X_i) \cap \mathcal{D}} D_j$. $\mathcal{X}_{Par(X_i)}$ is the Cartesian product of the domains of the state variables belonging to $Par(X_i)$ and $\mathcal{D}_{Par(X_i)}$ is the Cartesian product of the domains of the decision variables belonging to $Par(X_i)$. If X_i is a root of the DAG ($Par(X_i) = \emptyset$) we specify the a priori possibility degrees $\Pi(x_i)$ associated with each instance x_i of X_i .

Now if D_j is a decision node, its value d_j will be fixed by the decision maker. However, incoming links will specify which information is available when the decision is taken, and how the domain of D_j is restricted by the values of the variables in $Par(D_j)$ ⁵. Furthermore, we adopt the generally accepted *no forgetting assumption*, which implies that all the values of the variables that have been instantiated before d_j is chosen are still known at the time of the choice of d_j .

Solving a PID amounts to compute a *strategy*, allowing to maximise one of the possibilistic qualitative utilities. A strategy for a PID specifies the value of every decision variables as a function of all (or part of) the known values of the variables at the time the decision is taken.

Formally, a strategy is defined as :

Definition 1 (Strategy) In a PID, a strategy is a function $\delta : \mathcal{X} \rightarrow \mathcal{D}$ which can be represented as a set of local functions $\{\delta_j\}_{j \in 1 \dots p}$, $\delta_j : \Omega_j \rightarrow D_j$.

⁴ X_i will denote both the variable name and its domain, the meaning should be clear from the context. In the same way, \mathcal{X}_A , with $A \subseteq \{1, \dots, n\}$ will represent both a set of variables and the Cartesian product of their domains. The same holds for decision variables.

⁵ For sake of simplicity of exposition, we will assume in this paper that no restriction to the domain D_j will result from any instantiation of variables in $Par(D_j)$. However, this simplifying assumption is easy to relax, despite the fact that it induces cumbersome notations.

Ω_j denotes the cross product of the domains of all the variables (state and decision) instantiated before D_j . The variables instantiated before D_j (or before any state variable X_i) are called *ancestors* of D_j (of X_i). A strategy δ assigning an instantiation d to any global instantiation x of the state variables can thus be decomposed as a set of *local strategies* $\{\delta_j\}_{j \in 1 \dots p}$, δ_j specifying the value of decision variable D_j , for any possible instantiation of the variables in Ω_j . Notice that since Ω_j can contain decision variables, it is necessary to instantiate the decision variables in a suitable order (parents must be instantiated before their children). The fact that the diagram is a DAG guarantees that this order exists, and the above-mentioned assumption on the existence of a direct path comprising every decision variables guarantees its uniqueness.

Finally, for each utility node V_k , we prescribe ordinal values $\mu_k(x_{Par(V_k)}, d_{Par(V_k)})$ to every possible instantiations $(x_{Par(V_k)}, d_{Par(V_k)})$ of the parent variables of V_k . These values represent satisfaction degrees attached to the local instantiations of the parent variables. It is assumed, analogously to *Flexible Constraint Satisfaction Problems* [3], that the global satisfaction degree $\mu(x, d)$ associated with a global instantiation (x, d) of all variables (state and decision) can be computed as the *minimum* of the local satisfaction degrees : $\mu(x, d) = \min_{k=1 \dots q} \mu_k(x_{Par(V_k)}, d_{Par(V_k)})$.

Now, we are able to compute the optimistic and pessimistic utilities of a strategy δ . Once a strategy δ is chosen, the chance nodes form a possibilistic causal network and thus determine a unique *least specific* joint possibility distribution π_δ over chance nodes interpretations $x = \{x_1, \dots, x_n\}$. This joint possibility distribution can be computed by applying the *chain rule* [1] :

$$\pi_\delta(x) = \min_{k=1 \dots n} \Pi(x_k | x_{Par(X_k)}, d_{Par(X_k)}),$$

where $d = \delta(x)$. A global satisfaction degree $\mu_\delta(x)$ on state variables instantiations induced by a strategy can also be computed :

$$\mu_\delta(x) = \min_{l=1 \dots q} \mu_l(x_{Par(V_l)}, d_{Par(V_l)}).$$

Notice that $\pi_\delta(x)$ and $\mu_\delta(x)$ depend on the value of state variables only since a strategy δ uniquely defines the value $d = \delta(x)$ from x . It is then possible to compute the possibilistic optimistic and pessimistic qualitative expected utilities of a strategy δ :

$$\begin{aligned} u^*(\delta) &= \max \min_x \{\pi_\delta(x), \mu_\delta(x)\} \\ u_*(\delta) &= \min_x \max \{n(\pi_\delta(x)), \mu_\delta(x)\} \end{aligned} \quad (1)$$

Consider the example of Figure 1. The problem is to make a six-egg omelette from a five-egg one. The new egg can be fresh or rotten. There are two binary decision variable: break the egg in the omelette BIO ($BIO = yes$ if the egg is put in the omelette and $BIO = false$ if the egg is thrown away) ; break it apart in a cup BAC .

The state variables are the following. C : whether we have a cup to wash ($C = t$) or not ($C = f$). OF : whether we observe that the egg is fresh ($OF = t$) or rotten ($OF = f$) or we do not observe the egg ($OF = u$). F : the “real” state of the egg, fresh ($F = t$) or rotten ($F = f$). O : whether we get a six-egg omelette ($O = 6O$), a five-egg omelette ($O = 5O$) or no omelette at all ($O = nO$). S : whether an egg is spoiled ($S = t$) or not ($S = f$).

Let us take $L = \{0, \dots, 5\}$. The only a priori possibilities concern F , let us stand for $\Pi(F = t) = 5$ and $\Pi(F = f) = 3$. The

conditional possibilities are defined w.r.t. the natural meaning of the problem (here each possibility value is 0 or 5 but it is not always the case) :

$\Pi(C BAC)$	t	f	$\Pi(S F, BIO)$	(t, f)	(t, t)	(f, f)
t	5	0	t	5	0	0
f	0	5	f	0	5	5
u	0	0	u	0	0	5

$\Pi(O F, BIO)$	(t, t)	(f, t)	(\cdot, f)
t	5	0	0
f	0	5	0
u	0	0	5

$\Pi(O F, BIO)$	(\cdot, f)	(t, t)	(f, t)
$6O$	0	5	0
$5O$	5	0	0
nO	0	0	5

Utility functions μ_O, μ_S and μ_C are defined as follows, and for any global instantiation (x, d) , $\mu(x, d) = \min\{\mu_O(o), \mu_S(s), \mu_C(c)\}$.

O	$\mu_O(O)$	S	$\mu_S(S)$	C	$\mu_C(C)$
$6O$	5	f	5	t	5
$5O$	3	t	2	f	4
nO	0				

4 SOLUTION METHOD: DECISION TREE

Solving a PID amounts to finding an *optimal* strategy δ^* , i.e. one which maximises the (either optimistic or pessimistic) possibilistic utility. A PID can be unfolded into a *possibilistic decision tree*, using a method very similar to that of [6] for usual ID. A decision tree represents every possible scenarios of successive actions and observations, for a given ordering of decision variables⁶. In a possibilistic decision tree, branches issuing from a decision node are labelled with possible instantiations of the decision variable. Branches issuing from a chance node X_i are labelled with an instantiation of the variable and with its possibility degree computed from local transition possibilities together with the instantiations of the ancestors of X_i in the branch.

In order to build the decision tree, an ordering of the variables, compatible with the precedence constraints expressed by the DAG in Figure 1, must be chosen. Figure 2 shows a decision tree obtained by unfolding the PID in Figure 1 with the ordering $\{BAC, F, OF, BIO, O, S, C\}$, which is obviously compatible with the order induced by the DAG.

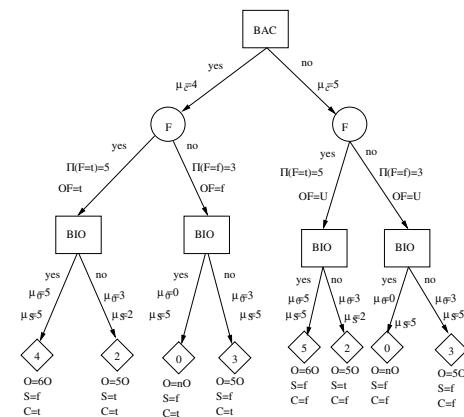


Figure 2. Decision tree associated with the example PID.

⁶ This is the reason why we choose to fix an ordering of decision variables in a PID.

The tree is then built by successively adding nodes, following the initial ordering. Branches issuing from the node are then added and labelled with the different possible values of the corresponding variable (either states or decisions). Note that in the decision tree of figure 2, after nodes *BAC* and *F* have been added, node *OF* is omitted since it has only one possible value, knowing the values of *BAC* and *F* (for example, in the leftmost branch *OF* is *true* since *F* is known to be *true* and *BAC* = *yes*). For the same reasons, nodes *C*, *O* and *S* are not displayed. Then, to each leaf of the decision tree is associated one particular interpretation of the variables. In figure 2, the leftmost branch corresponds to the instantiations $F = t, OF = t, O = 6O, S = f, C = t$ and $BAC = yes, BIO = yes$.

Then, to each leaf are attached both a possibility degree $\pi(x)$ and an aggregated satisfaction degree $\mu(x)$. Following the chain rule, $\pi(x)$ (resp. $\mu(x)$) is the minimum of the transition possibilities (resp. satisfaction degrees) involved in the branch from the root to the leaf. For example, the possibility degree of the leftmost branch of figure 2 is $\pi = \Pi(F = t) = 5$ and its utility is $\mu = \min(\mu_O(6O), \mu_C(t), \mu_S(f)) = 4$.

Now, in order to compute a possibilistic optimal strategy from the decision tree, we apply a *backwards induction* method similar to the one designed in [5]. First, note that choosing a strategy amounts to cut all branches issuing from each decision node, but one (corresponding to the action prescribed by the strategy, the values of the ancestor variables of the decision node being fixed in the branch). Figure 3 shows the pruned tree when strategy (*BAC* = *no*, *BIO* = *yes*) is chosen (note that this strategy is unconditional). In the optimistic case, the value of the leaf is $\min(\pi(x), \mu(x))$, hence the value of the leftmost leaf is 4. The utility of a strategy δ is then simply the maximum of the leaves values, corresponding to $u^*(\delta) = \max_{x \in \mathcal{X}} \min\{\pi_\delta(x), \mu_\delta(x)\}$. In Figure 3, the values of the two remaining leaves of the pruned tree are 5 and 0, thus the utility of the chosen strategy is 5. The pessimistic value of the same strategy corresponds to $u_*(\delta) = \min_{x \in \mathcal{X}} \max\{\pi_\delta(x), \mu_\delta(x)\}$. The remaining leaves take values $\max(0, 5) = 5$ and $\max(0, 3) = 3$ and the pessimistic utility is the minimum of these leaves values : $\min(5, 3) = 2$.

An optimal strategy can be computed by rolling the full decision tree backwards, keeping only one issuing branch for each decision node, and eliminating unobserved chance nodes:

- If all the children of a decision node already have an attached possibilistic utility, only its issuing branch with the highest value is kept. This highest value is attached to the decision node.
- If all the children of an unobserved chance node already have an attached possibilistic utility, it is replaced with a utility node containing its possibilistic (either optimistic or pessimistic) expected value.

These two steps are iterated until a possibilistic utility is attached to the root node. The remaining tree represents the optimal optimistic strategy, together with its utility. Figure 3 shows the optimal optimistic strategy computed with the omelette example. Note that the computed strategy (which is to break the egg directly in the omelette without using the cup) agrees with the intuition that since the a priori distribution on *F* makes *F* more likely than *noF* and since the

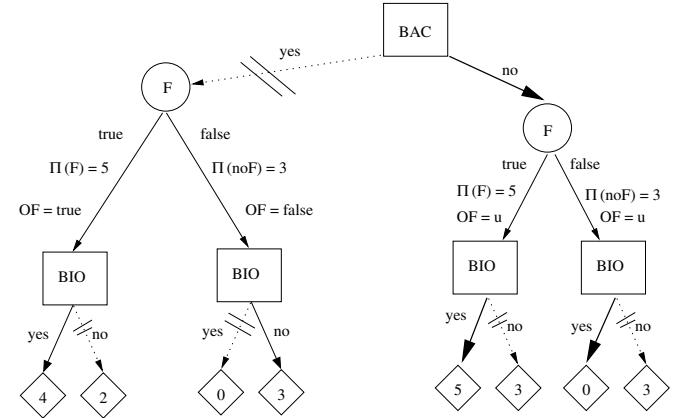


Figure 3. Optimal optimistic decision strategy computation.

decision maker is optimistic, he should not bother testing the freshness of the egg before using it.

5 COMPLEXITY RESULTS

The two basic questions which can be answered within the PID framework are :

1. Assessment of the possibilistic utility of a strategy δ .
2. Computation of an optimal strategy δ^* .

In this section we give the complexity of these problems, both for optimistic and pessimistic utilities. Proofs of Prop. 2, 3, 4 will only be sketched for sake of brevity since they are largely similar to that of Prop. 1. Note that NPO represent the class of optimisation problems “find $\alpha^* = \max_\alpha \dots$ ” for which the associated decision problem “is $\alpha^* = (\max_\alpha \dots) \geq \beta$?” is in NP.

Proposition 1 (Optimistic strategy evaluation (OSE))
A PID and a strategy δ being given, the problem of computing $u^*(\delta)$ is in NPO and is NP-hard.

Proof : OSE is a maximisation problem, since $u^*(\delta) = \max_x \min\{\pi_\delta(x), \mu_\delta(x)\}$. OSE is in NPO since it can be solved in polynomial time by the following non-deterministic algorithm, where step 2 can obviously be performed in polynomial time:

1. $\forall \alpha \in L$,
guess x ,
2. if $\min\{\pi_\delta(x), \mu_\delta(x)\} \geq \alpha$, return YES(α), else NO(α).
3. $u^*(\delta) = \max\{\alpha \text{ s.t. YES}(\alpha) \text{ holds}\}$.

Now, in order to show that OSE is NP-hard, we exhibit a polynomial transformation from 3-SAT to OSE. Let 3-SAT be defined as a set $\{v_1, \dots, v_n\}$ of variables and $\{\psi_1, \dots, \psi_m\}$ be a set of 3-clauses. Let us define the following PID, where $\mathcal{X} = \{X_1, \dots, X_n\}$ are binary variables, and the set of utility nodes is $\{r_0, r_1, \dots, r_m\}$. $\mathcal{D} = \{D_0\}$, where D_0 has domain $\{t, f\}$. Variables X_i have no parents and their unconditional possibility distributions are defined as $\Pi(X_i = t) = \Pi(X_i = f) = 1_L, \forall i$. There are arcs from X_i to r_j if and only if $v_i \in \text{Scope}(\psi_j)$. The local utility functions are the following : $\forall j > 0$, $\mu_j(x_{Par(r_j)}) = 1_L$ if $x_{Par(r_j)}$ satisfies ψ_j and 0_L else. D_0 is not connected to any other node. Then, if we fix δ

to $d_0 = t$ (δ is unconditional) and if we fix $\alpha = 1_L$, then obviously the utility of any δ is 1_L if and only if $\{\psi_1, \dots, \psi_m\}$ is satisfiable. So, this OSE solves 3-SAT for any strategy δ . Furthermore, the transformation is polynomial in time and space since utility tables have at most 2^3 elements (the ψ_j are 3-clauses). \square

Proposition 2 (Optimistic strategy optimisation (OSO))

A PID being given, the problem of computing an optimal optimistic policy δ^ and its utility $u^*(\delta)$ is in NPO and is NP-hard.*

Sketch of proof : Just notice that the OSO problem can be stated equivalently as the following maximisation problem : Find $U^* = \max_d \max_x \min\{\Pi(x|d), \mu(x, d)\}$. So, similarly to the OSE case, the problem will be here to guess interpretations (x, d) (instead of x only) for all fixed $\alpha \in L$, and to keep the “best one” : (x^*, d^*) . Thus, OSO is NPO. OSO is of course NP-hard, since OSE is. Note that δ^* is only defined in x^* ($\delta^*(x^*) = d^*$) and can take any value elsewhere. \square

Remark that it is obvious, from the proof of Prop. 2 that finding an optimal policy in an optimistic decision problem amounts to finding the branch with maximal leaf utility in the corresponding decision tree.

Proposition 3 (Pessimistic strategy evaluation (PSE))

A PID and a strategy δ being given, the problem of computing $u_(\delta)$ is in NPO and is NP-hard.*

Sketch of proof : PSE is a *minimisation* problem. $u_*(\delta) = \min_x \max\{n(\pi_\delta(x)), \mu_\delta(x)\}$. The same reasoning as for OSE can be used to show that PSE is NPO:

- 1. $\forall \alpha \in L$,
- 1. guess x ,
- 2. if $\max\{n(\pi_\delta(x)), \mu_\delta(x)\} \leq \alpha$, return YES(α), else NO(α).
- 3. $u^*(\delta) = \min\{\alpha \text{ s.t. YES}(\alpha) \text{ holds}\}$.

To show NP-hardness, we use the same transformation as for OSE, except that $\forall j > 0$, $\mu_j(x_{Par(r_j)}) = 0_L$ if $x_{Par(r_j)}$ satisfies ψ_j and 1_L else. Then the logic base will be satisfiable if and only if $u_* = 0_L$ (and not 1_L). \square

The problem, when solving a PSE, amounts to finding the branch in the pruned decision tree (a strategy being fixed) with *minimum* leaf utility.

The pessimistic strategy optimisation problem is more complex since the corresponding exploration of the (complete) decision tree alternates maximisation at decision nodes and minimisation at state nodes. The following result can be shown:

Proposition 4 (Pessimistic strategy optimisation (PSO))

A PID being given, the problem of computing an optimal pessimistic policy δ^ and its utility $u_*(\delta)$ is \sum_2^p -complete.*

Sketch of proof : We want to compute: $\max_\delta u_*(\delta) = \max_\delta \min_x \max\{n(\pi_\delta(x)), \mu_\delta(x)\}$, which we can write $\max_\delta u_*(\delta) = \max_d \min_x \max\{n(\Pi(x|d)), \mu(x, d)\}$. Let $\alpha \in L$ be given, “is $\max_\delta u_*(\delta) \leq \alpha$?” is equivalent to “is it true that $\exists x, \forall d, \max\{n(\Pi(x|d)), \mu(x, d)\} \leq \alpha$?” This question, of the form “is it true that $\exists x, \forall y, r(x, y)$?” is known to be \sum_2^p -complete. So, PSO can be solved by solving at most $|L| \sum_2^p$ -complete decision problems and is thus \sum_2^p -complete itself. \square

6 CONCLUSION

In this article we have described *Possibilistic Influence Diagrams* (PID), which allow to model in a compact form problems of sequential decision making under qualitative uncertainty in the framework of possibility theory. We have described a decision-tree based solution method for PID and we have given computational complexity results for several questions related to PID. Even though our complexity results seem negative in a sense, PID-related problems being at least NP-hard, these results have to be mitigated by comparing this complexity to that of structured stochastic MDP, shown to be EXP-hard [7], i.e. provably not in the polynomial hierarchy!

We now plan to work in the two following directions. First, even if we know that PID solving is at best NP-hard, we may look for more efficient solution methods than decision tree “brute force” exploration. We can either explore methods derived from stochastic ID [11, 12], such as *variable elimination* and to exploit specific features of PID. We can also try to exploit methods of exploration of *minimax* trees, since the decision tree obtained in the pessimistic case is a minimax tree. Another possible direction is to study more particularly structured *possibilistic Markov decision processes* [10] for which specific PID representations should be defined, representing explicitly the time-structure of the process (in the way 2-DBN are used to model structured MDP in [2]). This would lead to define structured versions of the possibilistic *value iteration* and *policy iteration* algorithms proposed in [10].

REFERENCES

- [1] S. Benferhat, D. Dubois, L. Garcia and H. Prade (2002). On the transformation between possibilistic logic bases and possibilistic causal networks. *Int. J. of Approximate Reasoning*, 29, 135-173.
- [2] C. Boutilier, R. Dearden and M. Goldszmidt (2000). Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121:1, 49-107.
- [3] D. Dubois, H. Fargier and H. Prade (1996). Possibility Theory in Constraint Satisfaction Problems: Handling Priority, Preference and Uncertainty, *Applied Intelligence* 6, 287-309.
- [4] D. Dubois and H. Prade (1995). Possibility theory as a basis for qualitative decision theory. *Proc. of IJCAI'95*, 1925-1930.
- [5] H. Fargier, J. Lang and R. Sabbadin (1998). Towards qualitative approaches to multi-stage decision making. *Int. J. of Approximate Reasoning*, 19, 441-471.
- [6] R.A. Howard and J.E. Matheson (1984). Influence Diagrams. *Readings on the Principles and Applications of Decision Analysis*, 2, 719-762. Menlo Park, CA.
- [7] C. Lusena, J. Goldsmith and M. Mundhenk (2001). Nonapproximability. Results for POMDP. *J. of Art. Int. Research*, 14, 83-103.
- [8] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- [9] M.L. Puterman (1994). *Markov Decision Processes*. John Wiley and Sons.
- [10] R. Sabbadin (2001). Possibilistic Markov decision processes. *Engineering Appl. of Artificial Intelligence*, 14, 287-300.
- [11] R.D. Schachter (1986). Evaluating influence diagrams. *Operations research*, 34(6), 871-882.
- [12] N. L. Zhang (1998). Probabilistic inference in influence diagrams. *Computational Intelligence*, 14:475-497/

Solving Optimization Problems with DLL

Enrico Giunchiglia¹ and Marco Maratea^{1, 2}

Abstract.

Propositional satisfiability (SAT) is a success story in Computer Science and Artificial Intelligence: SAT solvers are currently used to solve problems in many different application domains, including planning and formal verification. The main reason for this success is that modern SAT solvers can successfully deal with problems having millions of variables. All these solvers are based on the Davis-Logemann-Loveland procedure (DLL). DLL is a decision procedure: Given a formula φ , it returns whether φ is satisfiable or not. Further, DLL can be easily modified in order to return an assignment satisfying φ , assuming one exists. However, in many cases it is not enough to compute a satisfying assignment: Indeed, the returned assignment has also to be “optimal” in some sense, e.g., it has to minimize/maximize a given objective function.

In this paper we show that DLL can be very easily adapted in order to solve optimization problems like MAX-SAT and MIN-ONE. In particular these problems are solved by simply imposing an ordering on a set of literals, to be followed while branching. Other popular problems, like DISTANCE-SAT and WEIGHTED-MAX-SAT, can be solved in a similar way. We implemented these ideas in ZCHAFF and the experimental analysis show that the resulting system is competitive with respect to other state-of-the-art systems.

1 Introduction

Propositional satisfiability (SAT) is a success story in Computer Science and Artificial Intelligence: SAT solvers are currently used to solve problems in many different application domains, including planning [11], formal verification [6], and many others such as RNA folding, hand-writing recognition, graph isomorphism and sudoku problems. The main reason for this success is that modern SAT solvers can successfully deal with problems having millions of variables [5]. All these solvers are based on the Davis-Logemann-Loveland procedure (DLL) [9]. DLL is a decision procedure: Given a formula φ , it returns whether φ is satisfiable or not; Further, DLL can be easily modified in order to return an assignment satisfying φ , assuming one exists. However, in many cases it is not enough to compute a satisfying assignment: Indeed, the returned assignment has also to be “optimal” in some sense, e.g., it has to minimize/maximize a given objective function. MIN-ONE and MAX-SAT are two optimization versions of SAT. In MIN-ONE (resp. MIN-ONE \subseteq), given a satisfiable instance, the goal is to find a satisfying assignment in which the set of variables assigned to true is of minimal cardinality (resp. minimal). In MAX-SAT (resp. MAX-SAT \subseteq), given an unsatisfiable instance, the goal is to find an assignment in which the set of satisfied clauses is of maximal cardinality (resp. maximal). MIN-ONE and MAX-SAT problems have been studied, e.g., in [8, 16], and are particular cases of DISTANCE-SAT [3] and Pseudo-Boolean (see

e.g., [4]) problems. Our interest for MIN-ONE, MIN-ONE \subseteq , MAX-SAT and MAX-SAT \subseteq problems are related to their applications in the area of planning and formal verification, as briefly discussed in the rest of the paper. Besides these application domains, [8] shows that many important graph problems involving combinatorial optimization (such as Max-Cut, Max-Clique, and Min Vertex Cover) have linear-time reductions to MIN-ONE and MAX-SAT problems. Considering the procedures used to solve such problems, the standard approach is to modify DLL following a branch&bound schema. In the MIN-ONE case, whenever an assignment satisfying the input formula φ and with cost c_μ is found, search continues looking for another assignment satisfying φ but with a lower cost: The drawback of such approaches is that they may generate, at least in principle, all the assignments satisfying φ . Analogously for MAX-SAT problems.

In this paper we show that DLL can be very easily adapted in order to solve optimization problems like MIN-ONE, MAX-SAT and their variants. In particular these problems are solved by simply imposing an ordering on a set of literals, to be followed while branching. With such modification —differently from what happens for branch&bound approaches— the first discovered assignment satisfying the input formula, is guaranteed to be “optimal”: This assumes we are given a MIN-ONE/MIN-ONE \subseteq problem, but analogous considerations hold for the MAX-SAT/MAX-SAT \subseteq case. Other optimization problems, like DISTANCE-SAT and WEIGHTED-MAX-SAT, can be solved in a similar way. We implemented these ideas in ZCHAFF and the experimental analysis we conducted on MIN-ONE and MAX-SAT problems shows that:

- On MIN-ONE and MAX-SAT problems, our system is competitive with respect to other dedicated solvers and state-of-the-art systems used in the last Pseudo-Boolean evaluation [12].
- Considering MIN-ONE \subseteq problems, our solver is the fastest. In particular, our solver is much faster in solving MIN-ONE \subseteq instances than the corresponding MIN-ONE instances, while this is not the case for MAX-SAT \subseteq with respect to the MAX-SAT.
- Related to the previous point, comparing the cardinality $\#C$ (resp. $\#C\subseteq$) of the set of true variables returned when solving a MIN-ONE and a MIN-ONE \subseteq problem, we see that for most instances $\#C = \#C\subseteq$. Comparing the analogous values for MAX-SAT and MAX-SAT \subseteq , these are equal on all instances but three. Thus, provided we have an efficient solver for MIN-ONE \subseteq (resp. MAX-SAT \subseteq), it makes sense to use it also for MIN-ONE (resp. MAX-SAT) problems, at least for computing a “good” upper (resp. lower) bound.

The paper is structured as follows. In Section 2, we give the basic terminology and notation. We then present OPT-DLL, i.e., DLL modified in order to solve optimization problems (Section 3). How to model and solve optimization problems with OPT-DLL is showed in Section 4. The last two sections are devoted to the experimental analysis and the conclusions.

¹ DIST - Università di Genova

² Dipartimento di Matematica - Università della Calabria

2 Formal preliminaries

Given a set S , a relation " $\prec \subseteq S \times S$ " is a (strict or irreflexive) *partial order on S* if it has the following properties:

1. *Irreflexivity*: $a \not\prec a$, for each $a \in S$.
2. *Antisymmetry*: $a \prec b$ and $b \prec a$ implies $a = b$.
3. *Transitivity*: $a \prec b$ and $b \prec c$ implies $a \prec c$.

If for each two distinct $a, b \in S$, $a \prec b$ or $b \prec a$ then \prec is said to be a *total order*. It is common to call the pair S, \prec a *partially ordered set*.

Consider a set P of *variables*. A *literal* is a variable or the negation \bar{x} of a variable x . In the following, $\bar{\bar{x}}$ is the same as x .

A *clause* is a finite set of literals, and a *formula* is a finite set of clauses. An *assignment* is a consistent set of literals.

Consider an assignment μ and formula φ .

A literal l is *assigned by μ* if either l or \bar{l} is in μ . We say that μ

- is *total* if each variable in P is assigned by μ ;
- *satisfies* a formula φ if for each clause $C \in \varphi$, $C \cap \mu \neq \emptyset$.

A formula is *satisfiable* if there exists an assignment satisfying it.

Consider a partial order \prec on the set of total assignments. Intuitively speaking, $\mu' \prec \mu$ means that μ' is preferred to μ . Thus, for us, a *total assignment μ is optimal (with respect to \prec)* if

1. μ satisfies φ ; and
2. there is no total assignment μ' satisfying φ and with $\mu' \prec \mu$.

An *assignment μ is optimal (with respect to \prec)* if μ satisfies φ and there exists a total and optimal assignment extending μ .

3 OPT-DLL

As we anticipated in the introduction, OPT-DLL is like the standard DLL except for the heuristic. Given a formula φ , the basic idea of OPT-DLL is to first explore the search space where there can be a not yet ruled-out optimal assignment. In DLL, assuming that the current assignment is μ and that it is not the case that all the assignments extending μ are equally good, this amounts to knowing on which literal we have to branch.

To make these notions precise, consider a partially ordered set S, \prec in which S is a set of literals and \prec is such that for each literal $l \in S$, either $l \prec \bar{l}$ or $\bar{l} \prec l$: If \prec satisfies this condition, we say that \prec is *DLL-compatible (with respect to S)*. For example, the partial order on $\{\bar{x}_0, x_0, \bar{x}_1, x_1\}$ such that

$$\bar{x}_0 \prec x_0, \bar{x}_1 \prec x_1, \bar{x}_1 \prec \bar{x}_0, \quad (1)$$

is DLL-compatible with respect to $\{x_0, \bar{x}_0, x_1, \bar{x}_1\}$. Notice that the condition that for each $l \in S$, either $l \prec \bar{l}$ or $\bar{l} \prec l$ ensures that both $l \in S$ and $\bar{l} \in S$. Given this, the set S can be omitted from the specification of a DLL-compatible partially ordered set S, \prec .

The pseudo-code of OPT-DLL is represented in Figure 1, where:

- φ is a formula; μ is an assignment; \prec is a partial order DLL-compatible with respect to a set S of literals;
- $\text{assign}(l, \varphi)$ returns the formula obtained from φ by (i) deleting the clauses $C \in \varphi$ with $l \in C$, and (ii) deleting \bar{l} from the other clauses in φ ;
- $\text{ChooseLiteral}(\varphi, \mu, \prec)$ returns an unassigned literal l such that
 - if there exists a variable in S which is not assigned by μ , then each literal l' with $l' \prec l$ has to be assigned by μ , and

```

function OPT-DLL( $\varphi, \mu, \prec$ )
1 if ( $\emptyset \in \varphi$ ) return FALSE;
2 if ( $\varphi = \emptyset$ ) return  $\mu$ ;
3 if ( $\{\bar{l}\} \in \varphi$ ) return OPT-DLL( $\text{assign}(l, \varphi), \mu \cup \{l\}, \prec$ );
4  $l := \text{ChooseLiteral}(\varphi, \mu, \prec)$ ;
5  $v := \text{OPT-DLL}(\text{assign}(l, \varphi), \mu \cup \{l\}, \prec)$ ;
6 if ( $v \neq \text{FALSE}$ ) return  $v$ ;
7 return OPT-DLL( $\text{assign}(\bar{l}, \varphi), \mu \cup \{\bar{l}\}, \prec$ ).

```

Figure 1. The algorithm of OPT-DLL.

$-\$ is an arbitrary literal occurring in φ , otherwise.

OPT-DLL has to be invoked with φ and μ set to the input formula and the empty set respectively. It is easy to see that if the set S is empty, OPT-DLL is the same as DLL.

Assuming \prec is (1), OPT-DLL checks the existence of an assignment satisfying φ and extending one of $\{\bar{x}_1, \bar{x}_0\}$, $\{\bar{x}_1, x_0\}$, $\{x_1, \bar{x}_0\}$, $\{x_1, x_0\}$, following the order in which they are listed. Assuming x_1, x_0 are two variables encoding the values from 0 to 3, OPT-DLL will return

- an assignment with minimal corresponding value, if φ is satisfiable, and
- FALSE otherwise.

Assuming x_0, x_1 represent the actions of going by car and by plane respectively, (1) encodes the fact that we prefer to not perform these actions, and that not going by plane is preferred to not going by car. Consequently, OPT-DLL will first look for an assignment where both actions are false, then one in which we use only the car, then one in which we use only the plane, and only finally for assignments where we have to use both the car and the plane.

As the example makes clear, the partial order on the set S of literals induces a partial order on the set of total assignments. In the case of the example (1), if $\mu_0 = \{\bar{x}_0, \bar{x}_1\} \cup S_0$, $\mu_1 = \{\bar{x}_1, x_0\} \cup S_1$, $\mu_2 = \{x_1, \bar{x}_0\} \cup S_2$, $\mu_3 = \{x_1, x_0\} \cup S_3$ are four total assignments, we have

$$\mu_0 \prec \mu_1 \prec \mu_2 \prec \mu_3, \quad (2)$$

while, if μ and μ' are two total assignments assigning in the same way both x_0 and x_1 , $\mu \not\prec \mu'$, i.e., μ and μ' are equally good.

Assuming \prec is a given partial order on S , \prec is extended to the set of total assignments as follows: If μ and μ' are total assignments, $\mu \prec \mu'$ if and only if

1. there exists a literal $l \in S$ with $l \in \mu$ and $\bar{l} \in \mu'$; and
2. for each literal $l' \in S \cap (\mu' \setminus \mu)$, there exists a literal $l \in S \cap (\mu \setminus \mu')$ such that $l \prec l'$.

The first condition says that two total assignments are not in partial order if they assign in the same way the literals in S . The second condition says that μ is preferred to μ' if for each literal $l' \in S$ with $l' \in \mu'$ and $\bar{l}' \in \mu$, μ contains a literal $l \in S$ with $l \in \mu$ and $\bar{l} \in \mu'$, and l is preferred to l' (i.e., $l \prec l'$).

In the case of (1), the above definition leads to the partial order on the set of total assignment satisfying (2).

We can now state the formal result that OPT-DLL returns an optimal assignment, assuming the input formula is satisfiable.

Theorem 1 Let φ be a formula and \prec a DLL-compatible partial order on a set of literals. OPT-DLL($\varphi, \emptyset, \prec$) returns an optimal (with respect to \prec extended to the set of total assignments) assignment if φ is satisfiable, and returns FALSE otherwise.

4 Solving optimization problems with OPT-DLL

Considering OPT-DLL and our definition of optimality given in Section 2, it is clear that OPT-DLL can solve only those optimization problems in which the partial order on the set of total assignments can be obtained as the extension of a DLL-compatible partial order on a set of literals. Indeed, this is not always possible: Assuming we have only two variables x_0, x_1 , the total order on the set of total assignments $\{\bar{x}_1, \bar{x}_0\} \prec \{x_1, \bar{x}_0\} \prec \{x_1, x_0\} \prec \{\bar{x}_1, x_0\}$ is not obtainable as the result of the extension of a DLL-compatible partial order on a set of literals. Still, many important optimization problems can be easily modeled via a DLL-compatible partial order on a set of literals, and thus solved with OPT-DLL. Given a formula φ , we first consider MIN-ONE/MIN-ONE $_{\subseteq}$ and then MAX-SAT/MAX-SAT $_{\subseteq}$ problems: These problems are very interesting from an application perspective, as briefly described below. We also show how DISTANCE-SAT/DISTANCE-SAT $_{\subseteq}$ problems can be solved via OPT-DLL.

4.1 MIN-ONE and MIN-ONE $_{\subseteq}$

Let S be a subset of the set P of variables. Consider a satisfiable formula φ . Define $\text{MIN-ONE}^S(\varphi)$ (resp. $\text{MIN-ONE}_{\subseteq}^S(\varphi)$) to be the set of assignments μ satisfying φ and having $\mu \cap S$ of minimal cardinality (resp. minimal). It is clear that $\text{MIN-ONE}^S(\varphi) \subseteq \text{MIN-ONE}_{\subseteq}^S(\varphi)$.

In MIN-ONE (resp. MIN-ONE $_{\subseteq}$), the goal is to find an assignment μ in $\text{MIN-ONE}^P(\varphi)$ (resp. $\text{MIN-ONE}_{\subseteq}^P(\varphi)$). As pointed out in [8], MIN-ONE problems are interesting because various graph problems involving combinatorial optimization can be converted in linear time into them. In planning, if φ is a formula encoding a planning as satisfiability problem, any assignment satisfying φ corresponds to a sequence of (possibly parallel) actions achieving the goal: If S is the set of action variables in φ ,

1. if we want that as few as possible actions are executed, then we have to find an assignment in $\text{MIN-ONE}^S(\varphi)$;
2. if we want that no redundant sequence of (possibly parallel) actions is executed, then we have to find an assignment in $\text{MIN-ONE}_{\subseteq}^S(\varphi)$.

An assignment in $\text{MIN-ONE}_{\subseteq}^S(\varphi)$ can be easily computed via OPT-DLL, as stated by the following theorem, consequence of Theorem 1.

Theorem 2 Let S be a subset of P , and let φ be a formula. Let \prec be the DLL-compatible partial order such that $l \prec l'$ iff $l = \bar{x}$, $l' = x$, $x \in S$. If $\text{OPT-DLL}(\varphi, \emptyset, \prec)$ returns an assignment μ then $\mu \in \text{MIN-ONE}_{\subseteq}^S(\varphi)$. If $\text{OPT-DLL}(\varphi, \emptyset, \prec)$ returns FALSE then φ is unsatisfiable.

Intuitively speaking, DLL is forced to split first on the variables in S , assigning them to false.

An assignment in $\text{MIN-ONE}^S(\varphi)$ can be computed via OPT-DLL, assuming we have a formula encoding the objective function. This amounts to define a formula $\text{adder}(S)$ such that³

1. the only variables in common to $\text{adder}(S)$ and φ are those in S ;
2. if $n = \lceil \log_2(|S| + 1) \rceil$, $\text{adder}(S)$ contains n new variables b_{n-1}, \dots, b_0 ; and
3. for any total assignment μ to the variables in φ , there exists a unique total assignment ν to the variables in $\text{adder}(S)$ such that
 - (a) ν satisfies $\text{adder}(S)$;

³ The specification of $\text{adder}(S)$ will be used also in Section 4.3, where S is assumed to be an assignment.

- (b) μ and ν assign in the same way the variables in S , i.e., $\mu \cap S = \nu \cap S$;
- (c) $|\nu \cap S| = \sum_{i=0}^{n-1} \nu(b_i) \times 2^i$, where $\nu(b_i)$ is 1 if ν assigns b_i to true, and is 0 otherwise.

$\text{adder}(S)$ can be realized in polynomial time in many ways, see, e.g., [17]. If the above conditions are satisfied, we say that $\text{adder}(S)$ is an *adder of S with output b_{n-1}, \dots, b_0* .

Theorem 3 Let S be a subset of P . Let $\text{adder}(S)$ be an adder of S with output b_{n-1}, \dots, b_0 . Let φ be a formula. Let \prec be the DLL-compatible partial order on $\{b_{n-1}, \dots, b_0\}$ such that for each $i \in [0, n - 1]$, $\bar{b}_i \prec b_i$, and, if $i \neq 0$, $\bar{b}_i \prec \bar{b}_{i-1}$. If $\text{OPT-DLL}(\varphi \cup \text{adder}(S), \emptyset, \prec)$ returns an assignment μ then $\mu \cap P \in \text{MIN-ONE}^S(\varphi)$. If $\text{OPT-DLL}(\varphi, \emptyset, \prec)$ returns FALSE then φ is unsatisfiable.

Intuitively speaking, assuming no literal in $\{b_{n-1}, \dots, b_0\}$ is assigned as unit, OPT-DLL first explores the branches with the variables b_{n-1}, \dots, b_0 assigned to false; If all such branches fail, then OPT-DLL explores the branches in which b_{n-1}, \dots, b_1 are assigned to false while b_0 is assigned to true; If also these branches fail, then OPT-DLL explores the branches in which b_{n-1}, \dots, b_2, b_0 are assigned to false while b_1 is assigned to true; and so on and so forth.

4.2 MAX-SAT and MAX-SAT $_{\subseteq}$

Consider a formula φ . Let S be a subset of the clauses in φ . Intuitively speaking, we consider the problem of satisfying S and “as many as possible” clauses in $\varphi \setminus S$. Formally, define $\text{MAX-SAT}^S(\varphi)$ (resp. $\text{MAX-SAT}_{\subseteq}^S(\varphi)$) to be the set of assignments μ satisfying each clause in S and such that the set $\{C : C \in \varphi \setminus S, C \cap \mu \neq \emptyset\}$ is of maximal cardinality (resp. maximal). It is clear that $\text{MAX-SAT}^S(\varphi) \subseteq \text{MAX-SAT}_{\subseteq}^S(\varphi)$.

In MAX-SAT (resp. MAX-SAT $_{\subseteq}$), the goal is to find an assignment μ in $\text{MAX-SAT}^{\emptyset}(\varphi)$ (resp. $\text{MAX-SAT}_{\subseteq}^{\emptyset}(\varphi)$). The problem of determining an assignment in $\text{MAX-SAT}^S(\varphi)/\text{MAX-SAT}_{\subseteq}^S(\varphi)$ can be easily reduced to a MIN-ONE/MIN-ONE $_{\subseteq}$ problem by considering the formula $x(\varphi, S)$, obtained from φ by adding a newly created variable v_i to the i -th clause in $\varphi \setminus S$.

For example, if $\varphi = \{\{\bar{x}_0, \bar{x}_1\}, \{\bar{x}_0, x_1\}, \{x_0\}\}$ and $S = \{x_0\}$, $x(\varphi, S)$ is $\{\{v_1, \bar{x}_0, \bar{x}_1\}, \{v_2, \bar{x}_0, x_1\}, \{x_0\}\}$

Theorem 4 Let φ be a formula. Let S be a subset of φ . Let V be the set of variables in $x(\varphi, S)$ and not in φ . The following equalities hold:

$$\text{MAX-SAT}^S(\varphi) = \{\mu \cap P : \mu \in \text{MIN-ONE}^V(x(\varphi, S))\},$$

and

$$\text{MAX-SAT}_{\subseteq}^S(\varphi) = \{\mu \cap P : \mu \in \text{MIN-ONE}_{\subseteq}^V(x(\varphi, S))\}.$$

MAX-SAT is arguably the most studied problem in the SAT literature after the SAT problem itself. MAX-SAT $_{\subseteq}$ problems arise in many settings. For instance, in formal verification, if φ is a formula encoding an initial specification of a system, and if φ' encodes a refinement of the initial specification, a standard verification task is to prove that the refinement φ' is compatible with φ , i.e., that $\varphi \cup \varphi'$ is satisfiable. If $\varphi \cup \varphi'$ is unsatisfiable, one goal is to find “as large as possible” parts of the refinement which are consistent with the initial design: Such parts correspond to the assignments in $\text{MAX-SAT}_{\subseteq}^{\varphi}(\varphi \cup \varphi')$.

4.3 DISTANCE-SAT and DISTANCE-SAT \subseteq

DISTANCE-SAT [3] is another optimization problem in which, given a formula φ and an assignment μ , the goal is to find an assignment μ' satisfying φ and differing “as little as possible” from μ . Here we consider also its variant DISTANCE-SAT \subseteq . Formally: Let μ be an assignment. Define DISTANCE-SAT(φ, μ) (resp. DISTANCE-SAT \subseteq (φ, μ)) to be the set of assignments μ' satisfying φ and having the set $\{l : l \in \mu, \bar{l} \in \mu'\}$ of minimal cardinality (resp. minimal). It is clear that DISTANCE-SAT S (φ) \subseteq DISTANCE-SAT \subseteq (φ).

Theorem 5 Let φ be a formula. Let μ be an assignment. The following two facts hold:

1. Let \prec be the DLL-compatible partial order such that $l \prec \bar{l}$ if $l \in \mu$. If OPT-DLL($\varphi, \emptyset, \prec$) returns an assignment μ' then $\mu' \in$ DISTANCE-SAT \subseteq (φ, μ). If OPT-DLL($\varphi, \emptyset, \prec$) returns FALSE then φ is unsatisfiable.
2. Let adder(μ) be an adder of μ with output b_{n-1}, \dots, b_0 . Let \prec be the DLL-compatible partial order on $\{b_{n-1}, \dots, b_0\}$ such that for each $i \in [0, n-1]$, $b_i \prec \bar{b}_i$, and, if $i \neq 0$, $b_i \prec b_{i-1}$. If OPT-DLL($\varphi \cup$ adder(μ), \emptyset, \prec) returns an assignment μ' then $\mu' \cap P \in$ DISTANCE-SAT(φ, μ). If OPT-DLL($\varphi \cup$ adder(μ), \emptyset, \prec) returns FALSE then φ is unsatisfiable.

5 Implementation and experimental results

instance	#C	OPBDP	PBS4	MSAT+	optsat	#C \subseteq	optsat
1 bcomp5	39	0.95	8.87	0.4	7.08	85	0
2 bmax6	61	120.05	TIME	8.42	274.13	131	0
3 ibm2	940	TIME	TIME	19.73	TIME	1054	0.03
4 ibm3	6356	TIME	—	TIME	79.17	6422	0.65
5 gal8		MEM	—	SF	TIME	14207	2.87
6 3blocks	56	282.03	0.19	0.29	2.2	58	0.06
7 4blocksb	66	TIME	0.61	0.24	5.81	66	0.09
8 4blocks	108	TIME	115.96	50.94	TIME	116	0.39
9 large.c	265	TIME	0.94	0.96	1.5	272	0.3
10 large.d	431	TIME	—	7.71	99.75	443	1.1
11 log.a	135	TIME	TIME	1.39	7.53	135	0.04
12 log.b	138	TIME	TIME	8.99	TIME	138	0.04
13 rock.a	65	TIME	1.21	0.2	9.39	65	0.01
14 rock.b	69	TIME	0.14	0.27	5.9	69	0.01
15 r2b3.1	141	32.76	0.04	0.2	0.17	141	0.04
16 r2b3.2	138	67.14	0.03	0.08	0.14	138	0.03
17 r3b1.1	119	TIME	8.57	1.3	6.73	119	1.29
18 r3b1.2	126	TIME	7.09	0.82	8.49	126	0.21
19 r3b2.1	217	TIME	0.33	0.46	0.71	217	0.09
20 r3b2.2	206	TIME	0.25	0.53	0.73	206	0.08
21 qg1-8	64	TIME	81.46	31.06	85.67	64	6.87
22 qg2-7	49	75.03	0.23	0.27	0.72	49	0.16
23 qg2-8	64	MEM	54.26	21.83	29.56	64	20.83
24 qg3-8	64	19.62	0.24	0.1	0.61	64	0.02
25 qg4-9	81	TIME	53.12	19.36	250.69	81	0.2
26 qg5-11	121	MEM	0.25	0.43	0.86	121	0.15

Table 1. MIN-ONE (columns 3-7) and MIN-ONE \subseteq (columns 8-9) problems.

The implementation of a solver based on our ideas requires the modification of the heuristic of a DLL based SAT solver. In our case, we selected zCHAFF [13], the 2004 version. Such choice is motivated by our interest in solving large problems coming from applications, and by the fact that we already had some experience in hacking zCHAFF code.

instance	#C	BF	OPBDP	PBS4	MSAT+	optsat	#C \subseteq	optsat
1 barrel3	941	0.23	2.04	0.88	0.12	0.9	941	0.09
2 barrel4	2034	0.65	47.59	11.67	0.34	21.19	2034	1.03
3 barrel5	5382	21.42	MEM	MEM	24.01	177.11	5382	122.76
4 barrel6	8930	213.60	MEM	—	95.56	896.45	8930	1438.08
5 barrel7	13764	SF	MEM	—	285.55	435.46		TIME
6 Imult0	1205	0.39	13.05	1.45	0.16	7.35	1205	0
7 Imult2	3524	57.11	TIME	TIME	6.7	16.46	3524	0.1
8 Imult4	6068	261.74	MEM	—	35.34	98.05	6068	27.56
9 Imult6	8852	774.08	MEM	—	157.02	609.07		TIME
10 Imult8	11876	SF	MEM	—	297.32	704.08		TIME
11 qvar8	2272	0.62	MEM	17.67	2.95	36	2272	3.58
12 qvar10	5621	2.21	MEM	234.97	55.54	156.44	5621	48.25
13 qvar12	7334	6.2	MEM	—	36.8	74.49	7334	238.36
14 qvar14	9312	SF	MEM	—	117.25	815.66	9312	942.48
15 qvar16	6495	SF	MEM	—	51.33	117.31	6495	261.83
16 c432	1114	131.06	TIME	7.22	0.24	7.6	1114	0.74
17 c499	1869	TIME	TIME	100.41	0.8	4.59	1869	4.3
18 c880	2589	TIME	TIME	320.96	5.54	38.91	2589	16.72
19 c1355	3661	TIME	TIME	TIME	80.09	21.2	3661	30.86
20 c1908	5095	TIME	MEM	TIME	58.01	165.99	5095	129.95
21 c2670	6755	TIME	MEM	—	63.64	100.31	6755	359.52
22 c3540	9325	TIME	MEM	—	242.02	786.33		TIME
23 u-bw.a	3290	7.81	TIME	249	209.03	178.18	3288	0.04
24 u-bw.b		TIME	MEM	—	TIME	TIME	11487	87.61
25 u-log.a	5783	TIME	MEM	TIME	59.65	179.3	5782	1.35
26 u-log.b	6428	TIME	MEM	—	35.37	144.83	6428	19.68
27 u-log.c	9506	TIME	MEM	—	383.65	731.87	9506	76.89
28 u-rock.a	1691	13.29	TIME	41.29	206.56	6.26	1690	4.79

Table 2. MAX-SAT and MAX-SAT \subseteq problems, columns 3-8 and 9-10 resp.

For MIN-ONE \subseteq problems, the heuristics VSIDS of zCHAFF has been modified by simply selecting the unassigned literal l with highest VSIDS score, and then assigning the variable x in l to false. For MAX-SAT \subseteq , if there exists an unassigned literal l in $x(\varphi, S)$ and not in φ , the one with the highest VSIDS score is selected and the variable in it is assigned to false. Otherwise, the unassigned literal l with highest VSIDS score is selected and assigned to true. Analogous modifications have been done on VSIDS in order to solve MIN-ONE/MAX-SAT problems.

The solution of MIN-ONE/MAX-SAT problems also required the implementation of a function $adder(S)$ as specified in Section 4.1. As we already said, there are various ways to implement such function. We used the method described in [17] which takes linear time in the size of S . We call $optsat$ the resulting system.⁴ Beside the modification in the heuristic, we had also to modify zCHAFF pre-processor in order to disable the assignment of pure literals.

About the other solvers, we initially considered both dedicated solvers for MAX-SAT problems —like BF [7], MAXSATVER [18], WCSP [14]— and generic Pseudo-Boolean solvers —like OPBDP ver. 1.1.1 [4], PBS ver. 2.1 and ver. 4 [1], MSAT+ (abbreviation of MINISAT+) based on MINISAT ver. 1.13 [10]. MSAT+ was the solver able to prove unsatisfiability and optimality to a larger number of instances than all the other solvers that entered into the last Pseudo-Boolean evaluation [12]. Considering the dedicated solvers for MAX-SAT, we discarded MAXSATVER and WCSP after an initial analysis because they seem to be tailored for relatively small typically randomly generated problems, and are thus not suited to deal with problems coming from applications. About the Pseudo-Boolean solvers, we do not show the results for PBS ver 2.1 because it is almost always slower than ver 4.0.

Each solver has been run using its default settings. All the ex-

⁴ Available at <http://www.star.dist.unige.it/~marco/OPTSAT/>.

periments have been run on a Linux box equipped with a Pentium IV 2.4GHz processor and 1GB of RAM. The results for MIN-ONE/MIN-ONE \subseteq and MAX-SAT/MAX-SAT \subseteq problems are reported in Tables 1 and 2 respectively.⁵ CPU time is measured in seconds; timeout has been set to 1800 seconds. In the tables, “TIME” indicates that the solver does not solve the instance within the time limit; “MEM” indicates that the solver exceeds all the available memory; “SF” indicates that the solver exits abnormally; “–” indicates that the solver returns an incorrect result.

Considering the results for MIN-ONE problems in columns 4-7 of Table 1, we see that our solver **optsat** performs much better than all the other solvers except for MSAT+. OPBPD solves a few instances, PBS times out or outputs an incorrect result on large instances, our solver times out on four instances, while MSAT+ times out on 1 instance and on another instance it exits abnormally.

Considering MIN-ONE \subseteq problems, the results of our solver are shown in the last column of the table. Given that for any formula φ in the table $\emptyset \neq \text{MIN-ONE}(\varphi) \subseteq \text{MIN-ONE}_\subseteq(\varphi)$ and that it is not possible to codify MIN-ONE \subseteq problems in the other solvers we considered, it makes sense to compare the performances of our solver with the performances of the other solvers in columns 4-6. The first observation is that **optsat** is much faster than all the other systems: Almost all the problems are solved in less than 1s. Two other observations are in order:

1. Comparing **optsat** results in columns 7 and 9 we see that our solver is much faster in solving MIN-ONE \subseteq than MIN-ONE problems. This could have been expected given that handling MIN-ONE problems requires the encoding of adders counting the number of variables set to true, and many of the examples have more than a thousand variables (the “gal8” instance has ≥ 58000 variables).
2. comparing the cardinality $\#C_\subseteq$ in column 8 (resp. $\#C$ in column 3) of the optimal assignment returned by **optsat** when solving a MIN-ONE \subseteq (resp. MIN-ONE) problem, we see that for most instances $\#C = \#C_\subseteq$.

Considering the results for MAX-SAT problems in Table 2, we see that our solver **optsat** performs much better than all the other solvers, including the dedicated ones, except for MSAT+. In comparison to MSAT+, our solver is slower of a factor on most instances, but is also faster on some instances. As in the previous case, the performances of **optsat** on the same instances treated as MAX-SAT \subseteq problems are shown in the last column. Differently from the previous case, **optsat** is slower in solving MAX-SAT \subseteq than MAX-SAT problems except for the planning instances (rows (23-28)). We do not yet have a clear understanding of why this happens. We believe that this is related to the fact that for all the instances that we considered, $\#C/\#C_\subseteq$ (representing the cardinality of the set returned by **optsat** when solving a MAX-SAT/MAX-SAT \subseteq problem) are very close to number of clauses in the original SAT instance, but this is still subject of investigation.

6 Conclusions and future work

In this paper we showed that DLL can be used to solve optimization problems by simply imposing an ordering on the

⁵ In Table 1 (1-5) are Formal Verification instances ((1-2) from the Beijing’96 competition, (3-5) by Ofer Shtrichman); (6-14) are planning problems from SATPLAN; (15-20) are Data Encryption Standard problems; (21-26) are quasi group. In Table 2, (1-13) are Bounded Model Checking (BMC) problems used in the original BMC paper; (16-22) are miter-based circuit equivalence benchmarks by Joao Marques-Silva; (23-28) are planning problems from SATPLAN.

literals to be used while branching. We specifically considered MIN-ONE/MIN-ONE \subseteq /MAX-SAT/MAX-SAT \subseteq /DISTANCE-SAT/DISTANCE-SAT \subseteq problems, but it is clear that any optimization problem where the ordering on the set of total assignments can be obtained by extending a partial order on a set of literals can be handled by OPT-DLL. In particular, all the problems where the optimality condition is expressed via an objective function f , can be handled by OPT-DLL, provided we have a formula encoding the value of f . This is indeed the case, e.g., for WEIGHTED-MAX-SAT where the encoding is illustrated, e.g., in [17].

We implemented our ideas using zCHAFF as engine, and the encoding of [17] to solve MIN-ONE/MAX-SAT problems. The results are positive and encouraging. We believe that even better performances will be obtained by using MINISAT and, for MIN-ONE/MAX-SAT problems, using the encoding presented in [2, 15] of the objective function. Some of these encoding produce formulas of a bigger size, but they should lead to better performances of the back-end solver: See [2, 15] for more details.

ACKNOWLEDGEMENTS

This work is partially supported by MIUR.

REFERENCES

- [1] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah, ‘PBS: A backtrack search pseudo-Boolean solver’, in *Proc. SAT*, (2002).
- [2] Olivier Bailleux and Yacine Boufkhad, ‘Efficient CNF encoding of Boolean cardinality constraints.’, in *Proc. CP*, pp. 108–122, (2003).
- [3] Olivier Bailleux and Pierre Marquis, ‘Some Computational Aspects of DISTANCE-SAT’, *Journal of Automated Reasoning*, to appear, (2006).
- [4] P. Barth, ‘A Davis-Putnam enumeration algorithm for linear pseudo-Boolean optimization’, Technical report, Max Planck Institute for Computer Science, (1995). technical Report MPI-I-95-2-2003.
- [5] D. Le Berre and L. Simon, ‘Fifty-five solvers in Vancouver: The SAT 2004 competition’, in *Proc. SAT (Selected Papers)*, pp. 321–344, (2004).
- [6] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, ‘Symbolic model checking without BDDs’, in *Proc. TACAS*, (1999).
- [7] Brian Borchers and Judith Furman, ‘A two-phase exact algorithm for max-SAT and weighted max-SAT problems.’, *J. Comb. Optim.*, **2**(4), 299–306, (1998).
- [8] N. Creignou, S. Khanna, and M. Sudan, ‘Complexity classifications of Boolean constraint satisfaction problems’, *SIAM*, (2001).
- [9] M. Davis, G. Logemann, and D. Loveland, ‘A machine program for theorem proving’, *Journal of the ACM*, **5**(7), (1962).
- [10] Niklas Eén and Niklas Sörensson, ‘Translating pseudo-Boolean constraints into SAT’, *Journal on Satisfiability, Boolean Modeling and Computation*, (2006).
- [11] Henry Kautz and Bart Selman, ‘Planning as satisfiability’, in *Proc. ECAI*, pp. 359–363, (1992).
- [12] Vasco Miguel Manquinho and Olivier Roussel, ‘The first evaluation of pseudo-Boolean solvers (PB05)’, *Journal on Satisfiability, Boolean Modeling and Computation*, (2006).
- [13] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, ‘Chaff: Engineering an Efficient SAT Solver’, in *Proc. DAC*, (2001).
- [14] P. Meseguer S. D. Givry, J. Larrosa and T. Schiex, ‘Solving Max-SAT as weighted CSP’, pp. 363–376, (2003).
- [15] Carsten Sinz, ‘Towards an optimal cnf encoding of Boolean cardinality constraints.’, in *Proc. CP*, pp. 827–831, (2005).
- [16] John K. Slaney and Toby Walsh, ‘Phase transition behavior: from decision to optimization’, in *Proc. SAT*, (2002).
- [17] Joost P. Warners, ‘A linear-time transformation of linear inequalities into conjunctive normal form.’, *Inf. Process. Lett.*, **68**(2), 63–69, (1998).
- [18] Z. Xing and W. Zhang, ‘MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability’, *Artificial Intelligence*, **164**(1-2), 47–80, (2005).

Discovering Missing Background Knowledge in Ontology Matching

Fausto Giunchiglia and Pavel Shvaiko and Mikalai Yatskevich¹

Abstract. *Semantic matching* determines the mappings between the nodes of two graphs (e.g., ontologies) by computing *logical relations* (e.g., subsumption) holding among the nodes that correspond semantically to each other. We present an approach to deal with the lack of background knowledge in matching tasks by using semantic matching *iteratively*. Unlike previous approaches, where the missing axioms are manually declared before the matching starts, we propose a fully automated solution. The benefits of our approach are: (i) saving some of the pre-match efforts, (ii) improving the quality of match via iterations, and (iii) enabling the future reuse of the newly discovered knowledge. We evaluate the implemented system on large real-world test cases, thus, proving empirically the benefits of our approach.

1 INTRODUCTION

Match is a critical operator in many applications, e.g., AI, Semantic Web, WWW, e-commerce. It takes two graph-like structures, e.g., lightweight ontologies, such as Google and Looksmart², or business catalogs, such as UNSPSC and eCl@ss³, and produces a mapping between the nodes that correspond semantically to each other.

Many various solutions of match have been proposed so far, see for recent surveys [17, 3, 13, 9, 16]⁴. We focus on a schema-based solution, namely a matching system exploiting only the schema information, thus not considering instances. We follow the so-called *semantic matching* approach [6]. This approach is based on two key ideas. The first is that mappings are calculated between ontology entities by computing *logical relations* (e.g., equivalence, subsumption), instead of computing coefficients rating match quality in the $[0,1]$ range, as it is the case in the other approaches, e.g., [14, 5, 15]. The second idea is that the relations are determined by analyzing the *meaning* which is codified in the elements and the structures of ontologies. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the labels' intended meaning. This allows the translation of the matching problem into a propositional unsatisfiability problem, which can then be efficiently resolved using (sound and complete) state of the art propositional satisfiability (SAT) deciders, e.g., [2].

Recent industrial-strength evaluations of matching systems, see, e.g., [4, 1], show that lack of background knowledge, most often domain specific knowledge, is one of the key problems of matching systems these days. In fact, most state of the art systems, for the tasks of matching thousands of nodes show low values of *recall* ($\sim 30\%$), while with *toy* examples, the recall they demonstrated was most often around 90%. This paper addresses the problem of the missing background knowledge by using semantic matching iteratively. The contributions of the paper are: (i) the new automatic *iterative* semantic

matching algorithm, which provides such benefits as a better quality of match (recall), saving some of the pre-match efforts, enabling the future reuse of the newly discovered knowledge; (ii) the *quality evaluation* of the implemented system on large real-world test cases.

The rest of the paper is organized as follows. The semantic matching algorithm is briefly summarized in Section 2. Section 3 introduces the problem of the lack of background knowledge in matching and its possible solutions. Section 4 presents the iterative semantic matching algorithm and its details. Section 5 discusses experiments with matching lightweight ontologies. Section 6 reports some conclusions and outlines the future work.

2 SEMANTIC MATCHING

We focus on tree-like structures (e.g., Google, Looksmart, Yahoo!). *Concept of a label* is the propositional formula which stands for the set of documents that one would classify under a label it encodes. *Concept at a node* is the propositional formula which represents the set of documents which one would classify under a node, given that it has a certain label and that it is in a certain position in a tree.

The following relations can be discovered among the concepts at nodes of two ontologies: *equivalence* (=); *more/less general* (\sqsubseteq , \sqsupseteq); *disjointness* (\perp). When none of the relations holds, the special *idk* (I don't know) relation is returned. The relations are ordered according to decreasing binding strength, i.e., from the strongest (=) to the weakest (*idk*). *Semantic matching* is defined as follows: given two trees T1, T2 compute the $N_1 \times N_2$ *mapping elements*, $\langle ID_{i,j}, C_{1,i}, C_{2,j}, R' \rangle$, where $ID_{i,j}$ is a unique identifier of the given mapping element; $C_{1,i} \in T_1$; $i=1,\dots,N_1$; $C_{2,j} \in T_2$; $j=1,\dots,N_2$; R' is the strongest relation holding between the concepts at nodes $C_{1,i}, C_{2,j}$.

Let us summarize the semantic matching algorithm via a running example. We consider ontologies O1 and O2 shown in Figure 1, which are small parts of Google and Looksmart. The algorithm inputs two ontologies and outputs a set of mapping elements in four macro steps. The first two steps represent the pre-processing phase, while the third and the fourth steps are the element level and structure level matching respectively.

Step 1. For all labels L in two trees, compute concepts of labels.

Labels at nodes are viewed as concise descriptions of the documents that are stored under the nodes. The meaning of a label at a node is computed by inputting a label, by analyzing its real-world semantics, and by outputting a concept of the label, C_L . For example, by writing $C_{Hobbies_and_Interests}$ we move from the natural language ambiguous label *Hobbies_and_Interests* to the concept $C_{Hobbies_and_Interests}$, which codifies explicitly its intended meaning, namely the documents which are about hobbies and interests. Technically, based on WordNet (WN) [12] *senses*, concepts of labels are codified as propositional logical formulas, see [10] for details.

From now on, it is assumed that the propositional formula encoding the concept of label is the label itself. Numbers "1" and "2" are used as subscripts to distinguish between trees in which the given concept of label occurs, e.g., TOP_1 (belongs to O1) and TOP_2 (belongs to O2).

¹ Department of Information and Communication Technology, University of Trento, Povo, Trento, Italy. email: {fausto, pavel, yatskevi}@dit.unitn.it

² See, <http://www.google.com/Top/> and <http://www.looksmart.com/>

³ See, <http://www.unspsc.org/> and <http://www.eclass.de/>

⁴ See, www.OntologyMatching.org for a complete information on the topic.

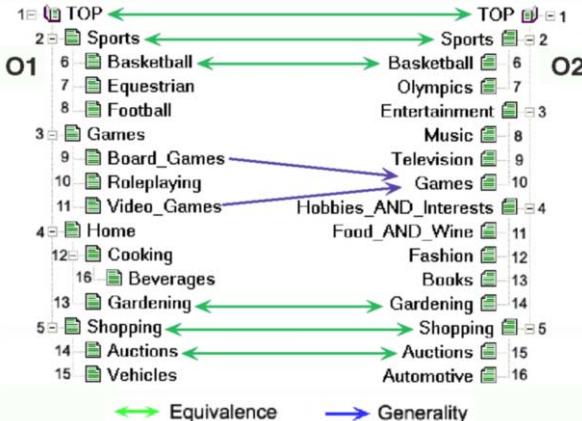


Figure 1. Parts of Google and Looksmart and some of the mappings

Step 2. For all nodes N in two trees, compute concepts at nodes.

We analyze the meaning of the *positions* of labels at nodes in a tree. By doing this concepts of labels are extended to concepts at nodes, C_N . This is required to capture the knowledge residing in the structure of a tree, namely the context in which the given concept at label occurs. Technically, concepts at nodes are written in the same propositional logical language as concepts of labels. For example, $C_{2_8} = TOP_2 \sqcap Entertainment_2 \sqcap Music_2$ stands for the concept describing all the documents about a particular kind of entertainment which is music.

Step 3. For all pairs of labels in two trees, compute relations among concepts of labels.

Relations between concepts of labels are computed by using a library of element level matchers, see Table 1. The first column contains the names of the matchers. The second col-

Table 1. Element level semantic matchers (Part 1)

Matcher name	Execution order	Approximation level	Matcher type	Schema info
WordNet	1	1	Sense-based	WordNet senses
Prefix	2	2	String-based	Labels
Suffix	3	2	String-based	Labels
Edit distance	4	2	String-based	Labels
Ngram	5	2	String-based	Labels

umn lists the order in which they are executed. The third column introduces the matchers' approximation level. The relations produced by a matcher with the first approximation level are always correct. For example, $Beverages_1$ can be found less general than $Food_2$. In fact, according to WordNet, *beverages* is hyponym (subordinate word) to *food*. Notice, in WordNet *beverages* has 1 sense, while *food* has 3 senses. Some sense filtering techniques are used to discard the irrelevant senses for the given context, see [10] for details. The relations produced by a matcher with the second approximation level are likely to be correct (e.g., *net* = *network*, but *hot* = *hotel* by *Prefix*). The WordNet matcher has two WordNet senses in input and computes equivalence, generality, and disjointness relations. String-based matchers have two labels as input. These compute only equivalence relations (e.g., equivalence holds if the weighted *distance* between the input strings is lower than a threshold), see [7]. String-based matchers are used iff WordNet fails to find a relation. The result of step 3 is a matrix of relations holding between atomic concepts of labels. A part of it, for the example of Figure 1, is shown in Table 2.

Table 2. cLabsMatrix: relations holding among atomic concepts of labels

	$Games_1$	$Board.Games_1$	$Beverages_1$
$Games_2$	=	\sqsupseteq	idk
$Food_2$	idk	idk	\sqsubseteq
$Wine_2$	idk	idk	\sqsubseteq

Step 4. For all pairs of nodes in two trees, compute relations among concepts at nodes. The tree matching problem is reformulated into a set of node matching problems, see Algorithm 1.

Algorithm 1 The tree matching algorithm

```

1: Node: struct of
2:   int nodeId;
3:   String label;
4:   String cLabel;
5:   String cNode;
6:   Node parent;
7:   AtomicConceptOfLabel[] ACOL;
8: AtomicConceptOfLabel: struct of
9:   int id;
10:  String token;
11:  String[] wnSenses;
12:  String[][] cLabsMatrix, cNodesMatrix;
30: String[] [] treeMatch(Tree of Nodes source, target)
31: Node sourceNode, targetNode;
32: int i, j;
33: String[] [] relMatrix;
34: String axioms, context1, context2;
35: cLabsMatrix = fillCLabMatrix(source, target);
50: for each sourceNode ∈ source do
51:   i = getNodeID(sourceNode);
52:   context1 = getCnodeFormula(sourceNode);
53:   for each targetNode ∈ target do
54:     j = getNodeID(targetNode);
55:     context2 = getCnodeFormula(targetNode);
80:     relMatrix = extractRelMatrix(cLabsMatrix, sourceNode, targetNode);
81:     axioms = mkAxioms(relMatrix);
82:     cNodesMatrix[i][j] = nodeMatch(axioms, context1, context2);
110: end for
111: end for
150: return cNodesMatrix;

```

In line 30, the *treeMatch* function inputs two trees of *Nodes* (*source* and *target*). Two loops are run over all the nodes of *source* and *target* trees in lines 50-111 and 53-110 in order to formulate all the node matching problems. Then, for each node matching problem, a pair of propositional formulas encoding concepts at nodes and relevant relations holding between concepts of labels are taken by using the *getCnodeFormula* and *extractRelMatrix* functions respectively. The former are memorized as *context₁* and *context₂* in lines 52 and 55. The latter are memorized in *relMatrix* in line 80. In order to reason about relations between concepts at nodes, the premises (*axioms*) are built in line 81. These are a conjunction of atomic concepts of labels which are related in *relMatrix*. Finally, in line 82, the relations holding between the concepts at nodes are calculated by *nodeMatch* and are reported in line 150 (*cNodesMatrix*). A part of this matrix for the example of Figure 1 is shown in Table 3.

Table 3. cNodesMatrix: relations holding among concepts at nodes⁴

	$C1_3$	$C1_4$	$C1_5$	$C1_9$	$C1_{10}$	$C1_{11}$
$C2_{10}$	=	idk	idk	\sqsupseteq	=	\sqsubseteq

nodeMatch translates each node matching problem into a propositional validity problem. Thus, we have to prove that $axioms \rightarrow rel(context_1, context_2)$ is valid. *axioms*, *context₁*, and *context₂* are as defined in the tree matching algorithm. *rel* is the logical relation that we want to prove holding between *context₁* and *context₂*. *nodeMatch* checks for sentence validity by proving that its negation is unsatisfiable. Thus, the algorithm uses, depending on a matching task, either ad hoc reasoning techniques, see [8], or standard DPLL-based SAT solvers, e.g., [2]. From the example in Figure 1, trying to prove that $C1_9$ is less general than $C2_{10}$, requires constructing the following formula: $((TOP_1 \leftrightarrow TOP_2) \wedge (Games_1 \leftrightarrow Games_2) \wedge (Games_1 \leftrightarrow Entertainment_2)^5 \wedge (Board.Games_1 \rightarrow Games_2)) \rightarrow ((TOP_1 \wedge Games_1 \wedge Board.Games_1) \rightarrow (TOP_2 \wedge Entertainment_2 \wedge Games_2))$. As it turns out, this formula is unsatisfiable, hence, the less generality holds.

⁵Notice, by applying element level matchers of Table 1 we can only determine the *idk* relation between *games* and *entertainment*. For example, in WordNet there is no direct lexical relation between *games* and *entertainment*. However, to simplify the presentation, we assume that it has been already determined that $Games_1 = Entertainment_2$. See Section 4 for the details of how the equivalence between the given concepts can be discovered.

3 LACK OF KNOWLEDGE

Recent industrial-strength evaluations of matching systems, see, e.g., [4, 1], show that lack of background knowledge, most often domain specific knowledge, is one of the key problems of matching systems these days. In fact, for example, should **PO** match **Post Office**, **Purchase Order**, or **Project Officer**? At present, most state of the art systems, for the tasks of matching thousands of nodes, perform not with such high values of *recall* ($\sim 30\%$) as in cases of toy examples, where the recall was most often around 90%. Also, contributing to this problem, [11] shows that complex matching solutions (requiring months of algorithms design and development) on big tasks may perform as badly as a baseline matcher (requiring one hour burden).

In order to understand better the above observations, let us consider a preliminary analytical comparative evaluation of some state of the art matching systems together with a baseline solution⁶ on three large real-world test cases. Table 4 provides some indicators of the test cases complexity.

Table 4. Some indicators of the complexity of the test cases

	#nodes	max depth	#labels per tree
Google vs. Looksmart	706/1081	11/16	1048/1715
Google vs. Yahoo	561/665	11/11	722/945
Yahoo vs. Looksmart	74/140	8/10	101/222

These test cases were taken from the OAEI-2005 ontology matching contest⁷. As match quality measures we focus here on recall which is a completeness measure. It varies in the $[0,1]$ range; the higher the value, the smaller is the set of correct mappings (true positives) which have not been found. The summarized evaluation results for all the three matching tasks are shown in Figure 2. Notice, the results for such matching systems as **OMAP**, **CMS**, **Dublin20**, **Falcon**, **FOAM**, **OLA**, and **ctxMatch2**, were taken from OAEI-2005, see [4], while evaluation results for the **baseline** matcher and **S-Match** were taken from [1]. As Figure 2 shows, none of the considered matching systems performs with a value of recall which is higher than 32%.

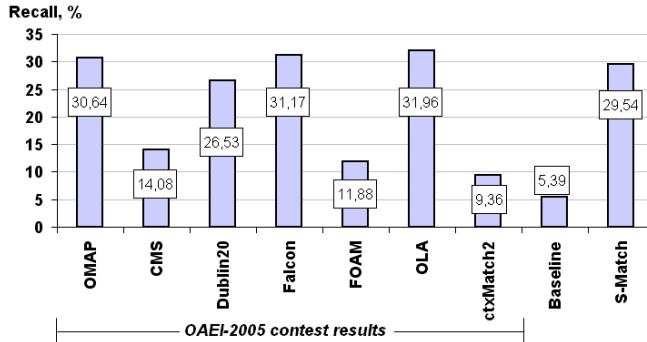


Figure 2. Analytical comparative evaluation

There are multiple strategies to attack the problem of the lack of background knowledge. One of the most often used methods so far is to declare the missing axioms manually as a pre-match effort. Some other plausible strategies include (i) extending stop word lists, (ii) expanding acronyms, (iii) reusing the previous match results, (iv) querying the web, (v) using (if available) domain specific sources of knowledge, and so on.

In this paper, we propose a fully automated solution to address the problem of the lack of knowledge by using semantic matching iteratively. The idea is to repeat *Step 3* and *Step 4* of the matching algorithm for some critical (hard) matching tasks. In particular, the result of SAT is analyzed. We identify critical points in the matching process, namely mapping elements with the *idk* relation where a stronger

⁶ This matcher does simple string comparison among sets of labels on the paths from nodes under consideration to the roots of the input trees, see [1].

⁷ See for details, <http://oaei.ontologymatching.org/2005>

relation (e.g., generality) should have taken place. We attack critical points by exploiting sophisticated element level matchers which use the *deep* information encoded in WordNet, e.g., its internal structure. Then, taking into account the newly discovered information as additional axioms, we re-run SAT solver on a critical task. Finally, if SAT returns *false*, we save the newly discovered knowledge, thereby enabling its future reuse.

4 ITERATIVE SEMANTIC MATCHING

We first discuss how the tree matching algorithm should be modified in order to suitably enable iterations. Then, we present the main building blocks of the iterative tree matching algorithm, namely, algorithms for critical points discovery and critical points resolution.

4.1 Iterative Tree Matching Algorithm

The iterative tree matching algorithm is shown as Algorithm 2. The numbers on the left indicate where the new code must be positioned in Algorithm 1.

Algorithm 2 A vanilla iterative tree matching algorithm

```

13: Boolean[ ][ ] cPointsMatrix;
100:   if (cPointsDiscovery(sourceNode, targetNode) == true) then
101:     cPointsMatrix[i][j] = true;
102:     ResolveCpoint(sourceNode, targetNode, context1, context2);
103:   end if

```

In line 13, we introduce **cPointsMatrix** which memorizes critical points. Semantic matching algorithm works in a top-down manner, and hence, mismatches among the top level classes of ontologies imply further mismatches between their descendants. Thus, the descendants should be processed only after the critical point at those top level nodes has been resolved. This is ensured by suitably positioning the new functions (enabling iterations) in a double loop of Algorithm 1. Hence, in line 100, we check with the help of **cPointsDiscovery** function if the nodes under consideration are the critical point. If they indeed represent the critical point, they are (memorized and) resolved by using the **ResolveCpoint** function (line 102). In the example of Figure 1, critical points which are determined are, for example, $\langle C_{12}, C_{23} \rangle$, $\langle C_{13}, C_{23} \rangle$, $\langle C_{14}, C_{24} \rangle$.

An updated **cNodesMatrix**, after running the iterative tree matching algorithm, is presented in Table 5. Comparing it with the non-iterative matching algorithm result, which is further reported in Table 7, we can see that having identified and resolved the $\langle C_{13}, C_{23} \rangle$ critical point, we also managed to discover the new correspondences, namely between C_{23} and C_{19} , C_{110} , C_{111} .

Table 5. Recomputed **cNodesMatrix**: relations among concepts at nodes

	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{19}	C_{110}	C_{111}
C_{21}	=	\sqsubseteq						
C_{23}	\sqsubseteq	=	=	idk	idk	\sqsubseteq	\sqsubseteq	\sqsubseteq

Having computed all the mapping elements for a given pair of ontologies, the identified critical relations are validated by a human user. In particular, user decides if the type of relation determined automatically is appropriate for the given pair of ontologies (e.g., is it appropriate that $Games_1 \leftrightarrow Entertainment_2$ or a weaker relation, namely $Games_1 \rightarrow Entertainment_2$, should have taken place ?), (s)he decides either to use this relation once (only for this pair of ontologies) or to save it in a domain specific oracle in order to enable its future reuse.

Finally, it is worth noting that iterative semantic matching algorithm amounts to *robustness* of the semantic matching. In fact, even if non-iterative semantic matching determines a (false) top level mismatch, this can be discovered and resolved by applying Algorithm 2. Thus, avoiding a further propagation of possible mismatches between the descendants of the initially mismatched top level nodes.

4.2 The Critical Points Discovery Algorithm

The algorithm for discovering critical points is based on the following intuitions:

- each *idk* relation in *cNodesMatrix* is potentially a critical point, but it is not always the case;
- since critical points arise due to lack of background knowledge, the clue is to check whether some other nodes located below the critical nodes (those representing a critical point) are related somehow. In case of a positive result the actual nodes are indeed the critical point; they represent a false alarm otherwise.

Algorithm 3 Critical points discovery algorithm

```

1: Boolean cPointsDiscovery(Node sourceNode, targetNode)
2: Node[ ] sDescendant, tDescendant;
3: ACOL sACOL, tACOL;
4: if (cNodesMatrix[sourceNode.nodeID][targetNode.nodeID]==“idk”)
then
5:   sDescendant = getSubTree(sourceNode);
6:   tDescendant = getSubTree(targetNode);
7:   for each sACOL ∈ sDescendant.ACOL do
8:     for each tACOL ∈ tDescendant.ACOL do
9:       if cLabsMatrix[sACOL.id][tACOL.id] != “idk” then
10:         return true;
11:       end if
12:     end for
13:   end for
14: else
15:   return false;
16: end if

```

Algorithm 3 formalizes these intuitions. In particular, the first condition mentioned above is checked in line 4. Verifying the second condition is more complicated. We call a relation holding between descendants of the potentially critical nodes a *support relation*. The support relation holds if there exists atomic concept of label (*sACOL*) in the descendants of *sourceNode* which is related in *cLabsMatrix* (by any semantic relation, except *idk*) to any atomic concept of label (*tACOL*) in the descendants of *targetNode*. This condition is checked in a double loop in lines 7-13. Finally, if both conditions are satisfied, the *cPointsDiscovery* function concludes that the nodes under consideration are the critical point (line 10). Under the given critical points discovery strategy, performing such a look up over the *cLabsMatrix* makes sense, obviously, only when *sourceNode* and *targetNode* are non-leaf nodes.

For example, suppose we want to match $C1_3$ and $C2_3$. Parts of *cLabsMatrix* (notice, the relations in this matrix were computed by applying element level matchers of Table 1) and *cNodesMatrix* with respect to the given matching task are shown in Table 6 and Table 7.

Table 6. *cLabsMatrix*: relations holding among atomic concepts of labels

	<i>TOP</i> ₁	<i>Games</i> ₁	<i>Board_Games</i> ₁
<i>TOP</i> ₂	=	<i>idk</i>	<i>idk</i>
<i>Entertainment</i> ₂	<i>idk</i>	<i>idk</i>	<i>idk</i>
<i>Games</i> ₂	<i>idk</i>	=	⊓

Table 7. *cNodesMatrix*: relations holding among concepts of nodes

	<i>C1</i> ₁	<i>C1</i> ₂	<i>C1</i> ₃	<i>C1</i> ₄	<i>C1</i> ₅	<i>C1</i> ₉	<i>C1</i> ₁₀	<i>C1</i> ₁₁
<i>C2</i> ₁	=	⊓	⊓	⊓	⊓	⊓	⊓	⊓
<i>C2</i> ₃	⊓	<i>idk</i>	<i>idk</i>	<i>idk</i>	<i>idk</i>	<i>idk</i>	<i>idk</i>	<i>idk</i>

In *cNodesMatrix* (Table 7) the relation between $C1_3$ and $C2_3$ is *idk*. In *cLabsMatrix* (Table 6) there is a support relation for the given matching problem, e.g., $Board_Games_1 \sqsubseteq Games_2$. Therefore, relation between $C1_3$ and $C2_3$ represents the critical point and we should reconsider the relation holding between $Games_1$ and $Entertainment_2$ in *cLabsMatrix*.

Finally, it is worth noting that this algorithm also properly handles nodes, which are indeed dissimilar, e.g., $C1_5$ and $C2_2$ are determined not to be the critical point.

4.3 The Critical Points Resolution Algorithm

Let us discuss how the critical points are resolved, see Algorithm 4.

Algorithm 4 Critical points resolution algorithm

```

1: ResolveCpoint(Node sourceNode, targetNode, String context1, context2)
2: String cRel;
3: String[ ] ExecutionList;
4: ACOL sACOL, tACOL;
5: for each sACOL ∈ sourceNode.ACOL do
6:   for each tACOL ∈ targetNode.ACOL do
7:     cRel = GetMLibRel(ExecutionList, sACOL.wnSenses, tACOL.wnSenses);
8:     cLabsMatrix[sACOL.id][tACOL.id] = cRel;
9:   end for
10: end for
11: cNodesMatrixUpdate(sourceNode, targetNode, context1, context2);

```

The *ResolveCpoint* function determines relations (*cRel*) for the critical points. Also, by exploiting the *cNodesMatrixUpdate* procedure, it updates accordingly *cNodesMatrix*. In particular, *ResolveCpoint* executes sophisticated element level matchers, see Table 8, over the atomic concepts of labels by using the *GetMLibRel* function (line 7). Matchers are applied following the order (*ExecutionList*) given in the second column of Table 8. These matchers have the third approximation level, which means that the relations they produce depend heavily on the context of the matching task. Also, execution times of them are much longer than of those of Table 1. Thus, they can not be applied in all the cases.

Table 8. Element level semantic matchers (Part 2)

Matcher name	Exec. order	Approx. level	Matcher type	Schema info
<i>Hierarchy Distance</i>	1	3	Sense-based	WN senses
<i>WN Gloss</i>	2	3	Gloss-based	WN senses
<i>Extended WN Gloss</i>	3	3	Gloss-based	WN senses
<i>Gloss Comparison</i>	4	3	Gloss-based	WN senses
<i>Extended Gloss Comparison</i>	5	3	Gloss-based	WN senses

For example, a *Hierarchy Distance* (*HD*) matcher computes the equivalence relation if the distance between two input senses in the WordNet hierarchy is less than a given threshold value (e.g., 3) and returns *idk* otherwise. According to WordNet, *games* and *entertainment* have a common ancestor, which is *diversion*. The distance between these concepts is 2 (1 more general link and 1 less general). Therefore, the *HD* matcher concludes that *Games*₁ is equivalent to *Entertainment*₂. If the *HD* matcher fails, which is not the case in our example, we apply a set of *gloss-based* matchers. These also have two WordNet senses in input and exploit techniques based on comparison of textual definitions (*glosses*) of the words whose senses are taken in input. They compute, depending on a particular matcher, the equivalence, more/less general relations. Due to lack of space, we give here only hints on how some gloss-based matchers work. For example, *WN Gloss* (*WNG*) counts the number of occurrences of the label of the source input sense in the gloss of the target input sense. If this number is lower than (equal to) a threshold, the less generality (due to a common pattern of defining terms in glosses through a more general term) is returned. *Gloss Comparison* (*GC*) counts the number of the same words in the glosses of the source and target input senses. If this number (of shared words) exceeds a threshold, the equivalence is returned. *Extended* gloss matchers are build in a straightforward way, by also considering glosses of the parent (children) nodes of the input senses in the WordNet *is-a* (*part-of*) hierarchy, see for details [7].

In line 8, we update *cLabsMatrix* with the critical relation, *cRel*, such that in all the further computations and for the current pair of nodes this relation is available. Finally, given the new axiom ($Games_1 \leftrightarrow Entertainment_2$) we recompute (line 11)⁸, by re-running SAT, the relation holding between the pair of critical nodes, thus determining that $C1_3 = C2_3$.

⁸ *cNodesMatrixUpdate* performs functionalities identical to those of lines 80-82 in Algorithm 1.

5 EVALUATION

In this section we present the quality evaluation of the iterative semantic matching algorithm. Due to lack of space we report here only what we have realized to be the most important findings.

Evaluation set-up. In our evaluation we have used three large real-world test cases, which were introduced in Table 4. As expert mappings for these test cases we used 2265 mappings acquired in [1]. By construction those expert mappings represent only true positives, thereby allowing us to estimate only the recall with them. To the best of our knowledge, at the moment, there are no large datasets as, for example, that one of Table 4, where available expert mappings allow measuring both precision⁹ and recall. Thus, in the following we focus mostly on analyzing the recall.

Two further observations. First is that higher values of recall can be obtained at the expense (lower values) of precision. Thus, in order to ensure a *fair* recall evaluation, before running tests on the matching tasks of Table 4, we have analyzed behavior of the iterative semantic matching on a number of test cases, e.g., course university catalogs¹⁰, where expert mappings allowed measuring both precision and recall. Matchers decreasing precision substantially in these tests were discarded from the further evaluation. In fact, for this reason we exclude from the further considerations the *Extended Gloss Comparison* matcher. The second observation is that using matchers of Table 8 exhaustively for all the tasks, hence, omitting the critical points discovery algorithm, also leads to a significant precision decrease, thus justifying usefulness of the cPointsDiscovery algorithm.

Evaluation results. The summarized evaluation results for all the three matching tasks of Table 4 are shown in Figure 3. In particular, it demonstrates contributions to the recall of matchers of Table 8 as well as of their combinations. The *Extended WN Gloss* matcher performed very poorly, i.e., contributing less than 1% to the recall, hence, we do not report its results in Figure 3. By using a combination of the *HD*, *WNG*, *GC* matchers we have improved S-Match recall results (29,5%) up to 46,1% within the iterative S-Match¹¹.

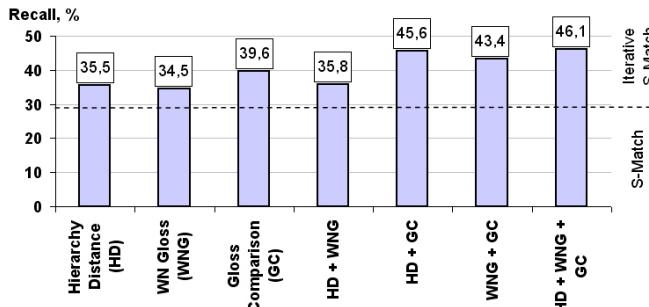


Figure 3. Evaluation results (absolute values)

Let us now consider *relative* characteristics of the iterative S-Match with respect to the non-iterative version, see the first row of Table 9 for a summary. The highest recall increase by using only a single matcher of Table 8 within the iterative S-Match was achieved by the *GC* matcher, namely 34% over the non-iterative S-Match. The best, in this sense, combination of two matchers is being that of the *HD* and *GC* matchers: recall increased by 54%. Finally, a combination of the *HD*, *WNG* and *GC* matchers resulted in the 56% recall increase with respect to the non-iterative S-Match.

Table 9 also reports values of thresholds used within the evaluation. These values were obtained based on the rationale behind designing matchers of Table 8 and their evaluation results.

⁹ Precision is a correctness measure; the higher the value, the smaller is the set of false positives which has been computed, see for details, e.g., [4].

¹⁰ See, <http://dit.unitn.it/~accord/Experimentaldesign.html> for a repository of test cases, expert mappings we have used.

¹¹ Note, this result should be considered as a complimentary one to the results of S-Match++ reported in [1], since they address separate problem spaces.

Table 9. Some element level matchers and their evaluation results

	HD	GC	HD + GC	HD + WNG + GC
Recall increase (relative), %	20	34	54	56
Threshold value	4	2	4 \ 2	4 \ 1 \ 2

Evaluation summary. The evaluation we have conducted shows that the problem of the lack of background knowledge is a hard one. In fact, as it turns out, not all the designed element level matchers can perform always well in real-world applications, as it might (mistakenly) seem from the toy evaluations. Also, new matchers are still needed, since, for example, we could discover that $\langle C_{14}, C_{24} \rangle$ is the critical point, however, we were unable to resolve it with the matchers of Table 8, namely to match *Home1* and *Hobbies AND Interests2*.

6 CONCLUSIONS

We have presented an automated approach to attack the problem of the lack of background knowledge by applying semantic matching iteratively. We implemented the proposed approach and evaluated it on large real-world (lightweight) ontology matching tasks with encouraging results. Future work proceeds at least in the following directions: (i) adapting the iterative semantic matching algorithm for handling leaf-node critical points, (ii) design and development of the new element level matchers, (iii) involving user within the matching process, where his/her input is maximally useful, (iv) conducting further large and extensive real-world evaluations.

REFERENCES

- [1] P. Avesani, F. Giunchiglia, and M. Yatskevich, ‘A large scale taxonomy mapping evaluation’, in *Proceedings of ISWC*, pp. 67 – 81, (2005).
- [2] D. Le Berre. A satisfiability library for Java. <http://www.sat4j.org/>.
- [3] A. Doan and A. Halevy, ‘Semantic integration research in the database community: A brief survey’, *AI Magazine*, (2005).
- [4] J. Euzenat, H. Stuckenschmidt, and M. Yatskevich, ‘Introduction to the ontology alignment evaluation 2005’, in *Proceedings of Integrating Ontologies workshop at K-CAP*, (2005).
- [5] J. Euzenat and P. Valtchev, ‘Similarity-based ontology alignment in OWL-lite’, in *Proceedings of ECAI*, pp. 333–337, (2004).
- [6] F. Giunchiglia and P. Shvaiko, ‘Semantic matching’, *Knowledge Engineering Review Journal*, (18(3)), 265–280, (2003).
- [7] F. Giunchiglia and M. Yatskevich, ‘Element level semantic matching’, in *Proceedings of the Meaning Coordination and Negotiation workshop at ISWC*, (2004).
- [8] F. Giunchiglia, M. Yatskevich, and E. Giunchiglia, ‘Efficient semantic matching’, in *Proceedings of ESWC*, pp. 272–289, (2005).
- [9] Y. Kalfoglou and M. Schorlemmer, ‘Ontology mapping: the state of the art’, *Knowledge Engineering Review Journal*, (18(1)), 1–31, (2003).
- [10] B. Magnini, L. Serafini, and M. Speranza, ‘Making explicit the semantics hidden in schema models’, in *Proceedings of the workshop on Human Language Technology for the Semantic Web and Web Services at ISWC*, (2003).
- [11] B. Magnini, M. Speranza, and C. Girardi, ‘A semantic-based approach to interoperability of classification hierarchies: Evaluation of linguistic techniques’, in *Proceedings of COLING*, (2004).
- [12] A.G. Miller, ‘WordNet: A lexical database for English’, *Communications of the ACM*, (38(11)), 39–41, (1995).
- [13] N. Noy, ‘Semantic Integration: A survey of ontology-based approaches’, *SIGMOD Record*, 33(4), 65–70, (2004).
- [14] N. Noy and M. Musen, ‘The PROMPT Suite: Interactive tools for ontology merging and mapping’, *International Journal of Human-Computer Studies*, (59(6)), 983–1024, (2003).
- [15] L. Palopoli, G. Terracina, and D. Ursino, ‘DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases’, *SPE*, 33(9), 847–884, (2003).
- [16] E. Rahm and P. Bernstein, ‘A survey of approaches to automatic schema matching’, *VLDB Journal*, (10(4)), 334–350, (2001).
- [17] P. Shvaiko and J. Euzenat, ‘A survey of schema-based matching approaches’, *Journal on Data Semantics*, (IV), 146–171, (2005).

Extracting MUSes

Éric GRÉGOIRE and Bertrand MAZURE and Cédric PIETTE ¹

Abstract. Minimally unsatisfiable subformulas (in short, MUSes) represent the smallest explanations for the inconsistency of SAT instances in terms of the number of involved clauses. Extracting MUSes can thus prove valuable because it circumscribes the sources of contradiction in an instance. In this paper, a new heuristic-based approach to approximate or compute MUSes is presented. It is shown that it often outperforms current competing ones.

1 INTRODUCTION

SAT is the NP-complete decision problem that consists in checking whether a set of Boolean clauses admits at least one truth assignment that satisfies all clauses. These last years, many researchers have focused on the more difficult task of extracting minimally unsatisfiable subformulas (in short, MUSes) of unsatisfiable SAT instances. Although this problem exhibits a high worst case complexity since e.g. checking whether a formula belongs to the set of MUSes of an inconsistent instance or not is in Σ_2^P [9], extracting MUSes can prove valuable because it circumscribes what is *wrong* with an instance. Indeed, many application domains like model-based diagnosis, knowledge-base validation or VLSI correctness checking, require such explanations to be delivered. When e.g. a knowledge-base is checked for consistency, we often prefer to know which clauses are actually contradicting one another, rather than just being told that the whole base is inconsistent.

Recently, several approaches have been proposed to approximate or compute MUSes. Unfortunately, they concern specific classes of clauses or they remain tractable for small instances only. Among them, let us mention Bruni's work [4], who has shown how a MUS can be extracted in polynomial time through linear programming techniques for clauses exhibiting a so-called integral point property. However, only restrictive classes of clauses obey such a property (mainly Horn, renamable Horn, extended Horn, balanced and matched ones). Let us also mention [5, 7, 10] as they are other important studies in the complexity and the algorithmic aspects of extracting MUSes for specific classes of clauses. In [3], Bruni has also proposed an approach that approximates MUSes by means of an adaptative search guided by clauses hardness. Zhang and Malik have described in [23] a way to extract MUSes by learning nogoods involved in the derivation of the empty clause by resolution. In [17], Lynce and Marques-Silva have proposed a complete and exhaustive technique to extract smallest MUSes. Oh and his co-authors have presented in [20] a Davis, Putnam, Logemann and Loveland DPLL-oriented approach that is based on a marked clause concept to allow one to approximate MUSes. Liffiton and Sakallah have shown how

MUSes can be computed through the dual concept of maximally satisfiable subsets [16].

In [19], a heuristic was also proposed to approximate MUSes, based on the experimental finding that clauses that are most often falsified during a failed local search often belong to MUSes. It has also been used to improve the performance of DPLL-like complete algorithms [6]. In this paper, a new variant and original extensions of this heuristic are studied. During the local search run, relevant parts of the neighborhood of the current truth assignment are explored in order to decide whether an unsatisfied clause during this local search should be actually counted or not. It is then extended in order to compute sets of MUSes. This new approach is shown to often outperform the current competing ones from an experimental point of view.

The paper is organized as follows. In the next section, the concept of MUS is presented formally. In section 3, a crucial notion of critical clause is introduced and analyzed. In section 4, the new approach to approximate or compute one MUS is presented. Extensive experimental results are given in section 5. Before we conclude, section 6 shows how the approach can be extended to compute sets of MUSes.

2 MINIMALLY UNSATISFIABLE SUBFORMULA (MUS)

Let \mathcal{L} be a standard Boolean logical language built on a finite set of Boolean variables, denoted a, b , etc. Formulas will be denoted using upper-case letters such as C . Sets of formulas will be represented using Greek letters like Γ or Σ . An interpretation is a truth assignment function that assigns values from $\{\text{true}, \text{false}\}$ to every Boolean variable. A formula is consistent or satisfiable when there is at least one interpretation that satisfies it, i.e. that makes it become *true*. An interpretation will be denoted by upper-case letters like I and will be represented by the set of literals that it satisfies. Actually, any formula in \mathcal{L} can be represented (while preserving satisfiability) using a set (interpreted as a conjunction) of clauses, where a clause is a finite disjunction of literals, where a literal is a Boolean variable that can be negated. SAT is the well-known NP-complete problem that consists in checking whether a set of Boolean clauses is satisfiable or not, i.e. whether there exists an interpretation that satisfies all clauses in the set or not.

When a SAT instance is unsatisfiable, it exhibits at least one minimally unsatisfiable subformula, in short one *MUS*.

Definition 1 A MUS Γ of a SAT instance Σ is a set of clauses s.t.

1. $\Gamma \subseteq \Sigma$
2. Γ is unsatisfiable
3. Every proper subset of Γ is satisfiable

Computing MUSes is a heavy computational task in the worst case. Indeed, checking whether a set of clauses is a MUS is DP-complete [21], and checking whether a formula belongs to the set of

¹ CRIL-CNRS & IRCICA,
 Université d'Artois, rue Jean Souvraz SP18, F-62307 Lens Cedex France
 E-mail: {gregoire,mazure,piette}@cril.univ-artois.fr

MUSes of an inconsistent instance or not, is in Σ_2^p [9]. Let us note that although the set of MUSes of an n -clauses instance is $C_n^{n/2}$ in the worst case, this number is often tractable in real-life situations. For example, in model-based diagnosis [13], based on experimental studies, it is often assumed that single faults occur most often, which is translated by a limited number of MUSes.

3 A NEW HEURISTIC TO DETECT MUSes

In [19], it is shown how local search can be helpful for approximating MUSes. The basic idea is that clauses that are often falsified during a failed local search for satisfiability belong most probably to MUSes, when the instance is actually unsatisfiable. When the score of a clause is the number of times it has been falsified during a failed local search (in short, failed LS), discriminating the clauses with a high score can deliver a good approximation of the set of MUSes. Such a heuristic has been studied in an extensive manner in [18, 19]. It has also been extended in several ways to address decision and optimisation problems that belong to higher levels of the polynomial hierarchy [11, 2, 12, 1].

In the following, we assume that the SAT instance is unsatisfiable. The above heuristic can require us to increment the score of clauses even when they do not actually belong to any MUS. Unless we solve the problem of finding MUSes itself, we can only rely on some heuristic indications about the extent to which a currently falsified clause could or could not belong to a MUS. In this respect, we claim that some relevant parts of the neighborhood of the current interpretation can be checked and provide more information about whether a currently falsified clause C should be counted or not. The idea is to take the structure of C into account and to increment the score of C only when it cannot be satisfied without conducting other clauses to be falsified in their turn. We shall see that this technique implements definitions that approximate a property that is intrinsic to clauses belonging to MUSes.

To illustrate this concept, let us use the following example. Let $\Delta = \{a \vee b \vee c, \neg a \vee b, \neg b \vee c, \neg c \vee a, \neg a \vee \neg b \vee \neg c\}$. Δ is unsatisfiable and is its own MUS. Let $I = \{a, b, c\}$ be an interpretation. Under this interpretation, only the clause $\neg a \vee \neg b \vee \neg c$ is falsified. In the following, the *once-satisfied* clause concept will prove useful.

Definition 2 A clause C is once-satisfied by an interpretation I iff only one literal of C is satisfied by I .

In the above example, the clauses $\neg a \vee b$, $\neg b \vee c$ and $\neg c \vee a$ are once-satisfied by $I = \{a, b, c\}$.

Definition 3 A clause C falsified by the interpretation I is critical w.r.t. I iff the opposite of every literal of C belongs to a clause that is once-satisfied by I . These once-satisfied clauses that are not tautological ones are called linked to C .

In the example, $\neg a \vee \neg b \vee \neg c$ is falsified by I and is critical w.r.t. I . Its related linked clauses are the once-satisfied ones $\neg a \vee b$, $\neg b \vee c$ and $\neg c \vee a$.

The role of these definitions is easily understood thanks to the following property.

Property 1 Let C be a critical clause w.r.t. the interpretation I , then any flip from I to I' that is such that C is satisfied under I' will conduct I' to falsify at least one clause that was satisfied by I .

In order to discriminate clauses belonging to MUSes, the idea is to increment the scores of critical clauses during the search, together with their linked (satisfied) clauses, rather than increment the scores of all falsified clauses. Indeed, this strategy exploits a property that is obeyed by clauses belonging to MUSes.

Property 2 Let I be an interpretation giving an optimal result for max-SAT on an inconsistent instance Σ . Then, any clause C of Σ falsified by I belongs to at least one MUS of Σ and is critical w.r.t. I . Moreover, at least one clause linked to C that is once-falsified by I also belongs to a MUS of Σ .

In this respect, a direct implementation of this technique would thus yield an approximation one in the sense that clauses and their linked ones are considered during the whole search, and not only at the best step of a max-SAT procedure. Such a technique can be easily grafted to a LS algorithm.

However, being a critical clause is neither a necessary nor a sufficient condition to belong to a MUS. As the following example illustrates, a critical clause w.r.t. an interpretation that is not an optimal one w.r.t. max-SAT for an unsatisfiable formula might not belong to a MUS. Let $\Delta = \{a \vee d, \neg a \vee \neg b, \neg d \vee e, f, \neg e \vee \neg f\}$. Clearly, Δ is consistent. $\neg e \vee \neg f$ is falsified by $I = \{a, b, d, e, f\}$ and is critical w.r.t. I . Moreover, a clause from a MUS that is falsified by a given interpretation I is not necessarily critical w.r.t. I , as the following example shows. Let $\Delta = \{a \vee d, b, \neg a \vee \neg b, \neg d \vee e, f, \neg e \vee \neg f\}$. Clearly, Δ is a minimal inconsistent set of clauses. $\neg a \vee \neg b$ is falsified by $I = \{a, b, d, e, f\}$. However, it is not critical w.r.t. I . Fortunately, the following property ensures that all clauses from a MUS can be scored by the heuristic.

Property 3 Let Γ be a MUS of Σ . For all clause $C \in \Gamma$, there exists an interpretation I s.t. C is critical w.r.t. I .

This property ensures that any clause that takes part in a MUS can be critical w.r.t. at least one interpretation. As such, this property does not guarantee that our scoring heuristic will allow us to exhibit all clauses belonging to MUSes. Indeed, it does not indicate that a LS run will necessarily increment the score of all such clauses at least once since LS does not necessarily visit all interpretations. However, the following property and its corollary provide us with a good indication that LS will probably visit interpretations where clauses belonging to MUSes are critical. Indeed, it is well-known that LS is in general attracted by local minima. Property 4 ensures that all falsified clauses are critical in local or global minima.

Definition 4 A local minimum is an interpretation s.t. no flip can increase the number of satisfied clauses. A global minimum (or max-SAT solution) is an interpretation delivering the maximal number of satisfied clauses.

Property 4 In (local or global) minima, all falsified clauses are critical.

A corollary even ensures that at least one clause per MUS is critical in such minima.

Corollary 1 In (local or global) minima, at least one clause per MUS is critical.

4 APPROXIMATING AND COMPUTING ONE MUS

In the following, it is shown that a meta-heuristic based on scoring critical clauses can be the cornerstone of a novel complete method to approximate or compute MUSes. Actually, due to implementation efficiency constraints, we update the scores of critical clauses only. Updating the scores of their linked clauses does not lead to dramatic performance improvements, at least w.r.t. our selected LS algorithm and tested benchmarks.

The main idea is as follows. Let Σ be an UNSAT instance. While LS fails to find a model of Σ (each time within a preset amount of computing resources), Σ is recorded on a stack and clauses that exhibit the lowest scores are removed from Σ . Then, the inconsistency of the last version of Σ for which LS has failed to find a model is checked using a complete search algorithm. If it is indeed inconsistent, then it is an upper-approximation of a MUS of the initial instance. Else, this inconsistency test is repeated on the next version of Σ from the stack until unsatisfiability is proved. Roughly, this algorithm is described in the following AOMUS procedure.

```
Procedure AOMUS ( $\Sigma$ ) // Approximate One MUS
begin
  stack :=  $\emptyset$  ;
  while (LS+Score ( $\Sigma$ ) fails to find a model)
    do
      push ( $\Sigma$ ) ;
       $\Sigma := \Sigma \setminus \text{LowestScore} (\Sigma)$  ;
    done
  repeat
     $\Sigma := \text{pop} ()$  ;
  until ( $\Sigma$  is UNSAT)
end
```

Then an exact MUS can be obtained by a step-by-step minimization of the upper-approximation until the remaining clauses are proved to form a MUS (see [14] for an alternative method). This process is called *fine-tune*. The order of tested clauses can be guided by the score of each clause.

```
Procedure fine-tune ( $\Sigma$ )
begin
  foreach clauses  $c \in \Sigma$  // sorted by scores
    If ( $\Sigma \setminus c$  is inconsistent) then  $\Sigma := \Sigma \setminus c$  ;
  done
end
```

The efficiency of this procedure directly depends on the quality of the upper-approximation. In the next section, experimental results show that the approximation delivered by AOMUS is often of a good quality, because a very small set of clauses is removed by the *fine-tune* step and in consequence a very small number of inconsistency tests are performed (when a clause belongs to the MUS, the test amounts to a consistency check).

Actually, we have refined this basic procedure in the following manner. Whenever a unique clause remains falsified during any of the LS runs, then we are sure that this clause belongs to all MUSes. We mark it as protected and it cannot be removed from Σ thereafter. When the remaining falsified clauses contain protected clauses only, they form one MUS: indeed, removing any one of these clauses will restore consistency. Moreover, when all clauses are protected and Σ is unsatisfiable, we are sure that Σ is a MUS and the *fine-tune* step

can be omitted. It appears that this refinement proves valuable for many instances, and allows a dramatic gain in the efficiency of the procedure. The OMUS algorithm includes the AOMUS procedure together with the *fine-tune* one with this refinement.

The parameters for these methods that were selected are as follows. Wsat [15] with the *Rnovelty+* option was chosen as the LS procedure. The following parameters were fine-tuned based on extensive tests on various benchmarks. After each flip of the LS, the score of critical clauses is increased by the number of their linked clauses. This technique allows us to take the length of critical clauses into account, since the number of linked clauses depends on the length of the critical clause in terms of the number of involved literals. Now, clauses whose score is lower than $(\text{min-score} + \frac{\#Flips}{\#Clauses})$ are dropped, where *min-score* is the lowest score for a clause of Σ ; *#Flips* and *#Clauses* are the number of performed flips and the number of clauses in Σ , respectively.

This procedure was tested extensively on various UNSAT instances from several difficult benchmarks from DIMACS [8] and from the annual SAT competitions [22], and compared with other published approaches to compute MUSes, as described in the next section.

5 EXPERIMENTAL RESULTS

All experiments have been conducted on Pentium IV, 3Ghz under linux Fedora Core 4. As our results show, this approximation delivers an exact result most of the time. Moreover, the *fine-tune* procedure ensures that a MUS is actually obtained. As most current approaches do not guarantee that the delivered inconsistent sets of clauses are actually MUSes, we provide both the results of applying our algorithm with and without the *fine-tune* routine. Without the *fine-tune* routine, the approach delivers upper-approximations of MUSes, and is called AOMUS (Approximate One MUS). However, on many instances, these approximations are actual MUSes. Moreover, it appeared very often that the last subformula for which LS failed to find a model was in fact unsatisfiable. Thus, in practice the last loop of the AOMUS algorithm often reduces to a unique inconsistency test. Let us stress that our approach is complete in the sense that it always delivers an approximate MUS for any UNSAT instance and a MUS when the *fine-tune* routine is run.

We compared our approach with an adaptation of AOMUS where *Scoring* is the basic heuristic of [19], which simply counts the number of times a clause is falsified. We also compared our approach with *zCore*, the core extractor of *zChaff* [23]. *zChaff* is currently one of the most efficient SAT solvers. We also ran Lynce and Marques-Silva's procedure [17], and took Bruni's [3] experimental results into account. For Bruni's technique, we only mention the experimental results obtained by the author, since this system is not available. Although a comparison with Bruni's technique is thus difficult to achieve from an experimental side, it appears that Bruni's technique has been experimented on small instances only. *zCore* proved competitive for single-MUS instances but failed to deliver good results when several MUSes are present. Indeed, *zCore* does not concentrate on finding one MUS, but on finding proofs of inconsistency. Not surprisingly, our approach proved more efficient than the similar one where *Scoring* is based on the heuristic from [19]. Most often, it proved to be more competitive than all the other considered techniques when very large and difficult multi-MUSes instances were considered. Noticeably, it was also the only technique to perform in a competitive way on all benchmarks. Let us stress that the Lynce-Silva's procedure computes the smallest MUS, that *zCore* delivers

Table 1. Experimental results: Approximate One MUS (AOMUS) and find One MUS (OMUS)

Instance	#var	#cla	Lynce&Silva [17]	Bruni [4]	zcore [23]	Scoring like [19]	AOMUS		OMUS	
			#cla	Time	#cla	Time	#cla	Time	#cla	Time
fpga10_11	220	1122	Time out	-	561	28.51	561	18.26	561	13.06
fpga10_12	240	1344	Time out	-	672	71.27	561	30.11	561	16.9
fpga10_13	260	1586	Time out	-	793	166.99	561	51.67	561	25.95
fpga10_15	300	2130	Time out	-	1065	570.3	561	128.05	561	44.18
fpga11_12	264	1476	Time out	-	738	112.53	738	66.8	738	65.49
fpga11_13	286	1742	Time out	-	871	504.97	738	180.66	738	56.71
fpga11_14	308	2030	Time out	-	1015	1565.6	738	415.32	738	69.55
fpga11_15	330	2340	Time out	-	738	568.79	738	52.14	738	304.4
aim100-1_6-no-2	100	160	53	224	54	0.05	53	0.268	53	0.38
aim100-2_0-no-1	100	200	Time out	-	19	0.09	19	0.216	19	0.19
aim200-1_6-no-3	200	320	Time out	-	86	0.07	83	0.37	83	0.44
aim200-2_0-no-3	200	400	Time out	-	37	0.23	37	0.39	37	0.49
aim50-1_6-no-4	50	80	20	1.18	20	0.04	20	0.163	20	0.16
aim50-2_0-no-4	50	100	21	3.49	21	0.14	21	0.208	21	0.22
2bitadd_10	590	1422	Time out	-	815	343.48	1212	42.752	806	189.47
barrel2	50	159	Time out	-	77	0.04	100	0.35	77	0.36
jnh10	100	850	Time out	-	161	0.88	128	9.35	79	42.25
jnh20	100	850	Time out	-	120	0.23	104	21.68	87	48.93
jnh5	100	850	Time out	-	125	0.39	140	12.653	88	46.2
jnh8	100	850	Time out	-	91	0.22	162	28.964	69	90.53
homer06	180	830	Time out	-	415	15.96	415	10.97	415	9.04
homer07	198	1012	Time out	-	506	21.6	415	12.59	415	10.67
homer08	216	1212	Time out	-	606	44.46	554	23.43	415	19.79
homer09	270	1920	Time out	-	960	141.48	415	93.19	504	60.9
homer10	360	3460	Time out	-	940	624.11	1614	148.27	503	466.94
homer11	220	1122	Time out	-	561	23.44	561	41.68	561	15.6
homer12	240	1344	Time out	-	672	76.19	708	25.92	564	41.03
homer13	260	1586	Time out	-	793	152.13	579	67.38	561	76.66
homer14	300	2130	Time out	-	1065	714.03	561	347.19	561	28.03
homer15	400	3840	Time out	-	677	247.84	561	1048.28	561	1104.13

More extensive results can be downloaded from <http://www.cril.univ-artois.fr/~piette/extratingMUS-comparison.pdf>

an approximation of a MUS, whereas our OMUS and AOMUS procedures deliver one exact and one approximate MUS, respectively. Moreover, it should be emphasized that MUSes that are discovered by the various approaches are not necessarily the same ones.

In Table 1, some typical experimental results are given. Except for Bruni's results which are just size results that we have extracted from [3], we provide both the experimental size of the discovered smallest inconsistent subsets, together with the CPU time in seconds to get them. Time-out indicates that no result has been obtained within 1 hour CPU time. For example, for the `homer14` instance, AOMUS delivered an approximate MUS made of 561 clauses within 28.03 s. Actually, this was an exact MUS, as it was found by OMUS in 30.64 s. Note that an AOMUS version based on [19] delivered the same result in 347.19 s. zCore delivered an approximate MUS made of 1065 clauses within 714 s. Actually, this approximate MUS was a superset of the MUS discovered by both AOMUS and OMUS. Also, it can be seen e.g. on the `fpga` benchmarks that AOMUS (i.e. our approach without the `fine-tune` procedure) delivered smaller inconsistent subsets than any other considered method, most often. Let us also emphasize that even on small instances like the `aim` ones, OMUS proved very competitive, as well.

6 APPROXIMATING THE SET OF MUSes

Based on the OMUS procedure, we now address the problem of computing the set of MUSes of unsatisfiable instances, also called *clutter* by Bruni [4]. Since a MUS can be “broken” by removing one of its clauses, a naive approach consists in removing one clause of a MUS after this latter one has been discovered by the OMUS procedure, and then in iterating the process. Such an approach would deliver the right result when any pair of MUSes exhibits an empty intersection. However, MUSes can have non-empty intersections. Ac-

cordingly, when we remove a clause from a MUS, we actually break all MUSes containing it. To prevent this drawback from occurring as much as possible, we should prefer dropping clauses that belong to a minimal number of MUSes. Accordingly, we have investigated the following heuristic.

As max-SAT is intended to deliver a minimal number of unsatisfied clauses, the remaining unsatisfied clauses in a max-SAT solution must belong to intersections of MUSes as much as possible. Accordingly, for each clause, we record the minimum number of clauses that have been falsified at the same time during a failed LS. After a MUS is detected, the clause in the MUS with the lowest score is removed from the instance.

Clearly, such an approach (that we note ASMUS (Approximate Set of MUS)) is incomplete. However, it delivers very good results with respect to current existing approaches, as illustrated by our experimental investigations summarized in Table 2. For these experiments, the time-out was set to 20000 s. Aleat X_Y_Z instances are standard generated (unsatisfiable) random ones, with X variables and Y clauses. XAIM $\alpha\beta$ _ Z instances are the mere concatenations of X AIM $\alpha\beta$ instances, where $\alpha = \frac{Y}{X}$ and $\beta = \frac{Z}{Y}$.

We have compared the ASMUS method with the complete algorithm proposed in [16] from an experimental point of view. Table 2 shows that both approaches appear to deliver the *exact* sets of MUSes on the simple “aim” benchmarks, using similar run-times. On more difficult instances like Aleat30_75_*, ASMUS almost extracts all MUSes and its computation time is in general better than the complete method one.

Moreover, Liffiton and Sakallah's algorithm can get into trouble for larger instances, since a CNF formula can exhibit an exponential number of MUSes and since their approach aims at computing all MUSes individually, only after having computed all maximally sat-

Table 2. Finding as many MUSes as possible.

Instance	#var	#cla	L.&S. [16]		ASMUS	
			#MUS	Time	#MUS	Time
aim100-1_6-no-1	100	160	1	0.18	1	0.31
aim200-1_6-no-1	200	320	1	0.14	1	0.68
aim200-1_6-no-2	200	320	2	0.22	2	0.76
aim200-2_0-no-3	200	400	1	0.12	1	0.56
aim200-2_0-no-4	200	400	2	0.26	2	0.88
Aleat20_70_1	20	70	127510	6.9	6	4.9
Aleat20_70_2	20	70	114948	10.8	13	8.7
Aleat30_75_1	30	75	11	59.82	7	2.2
Aleat30_75_2	30	75	9	26.84	8	2.9
Aleat30_75_3	30	75	10	12.84	10	3.7
Aleat50_218_1000	50	218	Time out		67	173
Aleat50_218_100	50	218	Time out		39	126
2AIM100_160	100	160	2	0.21	2	0.69
2AIM400_640	400	640	2	14.9	2	3.1
3AIM150_240	150	240	3	73.84	3	1.46
4AIM200_320	200	320	Time out		4	2.82
dp02u01	213	376	Time out		14	26.12
Homer06	180	830	Time out		2	17.47

isifiable subformulas, which can be intractable. On the opposite, our approximation technique does not suffer from such a drawback and exhibits an anytime property since MUSes are directly computed one after the other. For example, let us consider the Aleat20_70_2 random instance. It exhibits 70 clauses and these constraints form more than 114 000 MUSes. Due to the very small size of this formula, [16] has been able to compute all MUSes. For larger instances involving many MUSes, like dp02u01 (213 atoms, 376 clauses), the set of MUSes could not be computed within 20000 s., while our approach extracted 14 MUSes in 26 s.

7 CONCLUSION

In this paper, thanks to an original concept of critical clauses, a novel meta-heuristic-based approach to compute MUSes in SAT instances has been introduced. As our experimental results on difficult benchmarks illustrate it, the approach proves to be viable and often more competitive than previously published ones. The meta-heuristic is based on the intuitive idea that the most often falsified constraints during a failed local search are often the actual unsatisfiable ones. This idea has been refined to take the falsification propagation effect of these constraints. We believe that such a meta-heuristic could be applied to various difficult decision and optimisation problems. We plan to explore this in the near future.

REFERENCES

- [1] F. Boussemart, F. Hémery, C. Lecoutre, and L. Saïs, ‘Boosting systematic search by weighting constraints’, in *European Conference on Artificial Intelligence (ECAI’04)*, pp. 146–150, (2004).
- [2] L. Brisoux, É. Grégoire, and L. Saïs, ‘Checking depth-limited consistency and inconsistency in knowledge-based systems’, *International Journal of Intelligent Systems*, **16**(3), 333–360, (2001).
- [3] R. Bruni, ‘Approximating minimal unsatisfiable subformulae by means of adaptive core search’, *Discrete Applied Mathematics*, **130**(2), 85–100, (2003).
- [4] R. Bruni, ‘On exact selection of minimally unsatisfiable subformulae’, *Annals of Mathematics and Artificial Intelligence*, **43**(1), 35–50, (2005).
- [5] H.K. Büning, ‘On subclasses of minimal unsatisfiable formulas’, *Discrete Applied Mathematics*, **107**(1–3), 83–98, (2000).
- [6] M. Davis, G. Logemann, and D. Loveland, ‘A machine program for theorem proving’, *Journal of the Association for Computing Machinery*, **5**(7), 394–397, (1962).
- [7] G. Davydov, I. Davydova, and H.K. Büning, ‘An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF’, *Annals of Mathematics and Artificial Intelligence*, **23**(3–4), 229–245, (1998).
- [8] DIMACS. Benchmarks on SAT. <ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/>.
- [9] T. Eiter and G. Gottlob, ‘On the complexity of propositional knowledge base revision, updates and counterfactual’, *Artificial Intelligence*, **57**, 227–270, (1992).
- [10] H. Fleischner, O. Kullman, and S. Szeider, ‘Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference’, *Theoretical Computer Science*, **289**(1), 503–516, (2002).
- [11] É. Grégoire and D. Ansart, ‘Overcoming the christmas tree syndrome’, *International Journal on Artificial Intelligence Tools (IJAIT)*, **9**(2), 97–111, (2000).
- [12] É. Grégoire, B. Mazure, and L. Saïs, ‘Using failed local search for SAT as an oracle for tackling harder A.I. problems more efficiently’, in *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AMSA’02)*, LNCS 2443, pp. 51–60, (2002).
- [13] *Readings in Model-Based Diagnosis*, eds., Console L. Hamscher W. and de Kleer J., Morgan Kaufmann Publishers Inc, 1992.
- [14] J. Huang, ‘Mup: A minimal unsatisfiability prover’, in *Asia and South Pacific Design Automation Conference (ASP-DAC’05)*, pp. 432–437, (2005).
- [15] H. Kautz, B. Selman, and D. McAllester, ‘Walksat in the SAT 2004 competition’, in *International Conference on Theory and Applications of Satisfiability Testing (SAT’04)*, (2004).
- [16] M.H. Liffiton and K.A. Sakallah, ‘On finding all minimally unsatisfiable subformulas’, in *International Conference on Theory and Applications of Satisfiability Testing (SAT’05)*, pp. 173–186, (2005).
- [17] I. Lynce and J. Marques-Silva, ‘On computing minimum unsatisfiable cores’, in *International Conference on Theory and Applications of Satisfiability Testing (SAT’04)*, (2004).
- [18] B. Mazure, L. Saïs, and É. Grégoire, ‘A powerful heuristic to locate inconsistent kernels in knowledge-based systems’, in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU’96)*, pp. 1265–1269, (1996).
- [19] B. Mazure, L. Saïs, and É. Grégoire, ‘Boosting complete techniques thanks to local search’, *Annals of Mathematics and Artificial Intelligence*, **22**, 319–331, (1998).
- [20] Y. Oh, M.N. Mneimneh, Z.S. Andraus, K.A. Sakallah, and I.L. Markov, ‘Amuse: a minimally-unsatisfiable subformula extractor’, in *Design Automation Conference (DAC’04)*, pp. 518–523, (2004).
- [21] C.H. Papadimitriou and D. Wolfe, ‘The complexity of facets resolved’, *Journal of Computer and System Sciences*, **37**(1), 2–13, (1988).
- [22] SATLIB. Benchmarks on SAT. <http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html>.
- [23] L. Zhang and S. Malik, ‘Extracting small unsatisfiable cores from unsatisfiable Boolean formula’, in *International Conference on Theory and Applications of Satisfiability Testing (SAT’03)*, (2003).

ANNEX: PROOFS

Proof of Property 1 If C is critical w.r.t. I then for each literal l of C , $\exists C'$ s.t. C' is once-satisfied by I and \bar{l} belongs to C' . C is falsified by I , thus l is false w.r.t. I and \bar{l} is true w.r.t. I . \bar{l} is the only literal of C' satisfied by I . Accordingly if the value of l is reversed then C' becomes falsified. \square

Proof of Property 2 Any clause falsified by I belongs to a MUS of Σ because I is optimal w.r.t. the number of satisfied clauses and at least one clause of each MUS cannot be satisfied by I . The fact that any clause falsified by I is critical is proved thanks to property 4 since I is a global minimum. I is optimal w.r.t. the number of satisfied clauses, thus at most one clause per MUS is falsified. Also, if one flip allows us to satisfy one of these clauses, another clause of the MUS becomes falsified. Accordingly, at least one once-satisfied clause linked to a clause falsified by I belongs to a MUS of Σ . \square

Proof of Property 3 Let Γ be a MUS of Σ and C be a clause of Γ . By definition of a MUS, $\Gamma \setminus \{C\}$ is satisfiable. Let M be a model of $\Gamma \setminus \{C\}$. Let us prove that C is critical w.r.t. M . First, C is falsified. Indeed, if C were not falsified then Γ would exhibit a model M , which is impossible because Γ is a MUS. Second, C is critical. Indeed, if any variable occurring in C is flipped w.r.t. M , then at least one clause of Γ becomes unsatisfied since Γ is unsatisfiable. That means that this new unsatisfied clause was once-satisfied and linked to C . Accordingly, C is critical w.r.t. M . \square

Proof of Property 4 If a variable occurring in a falsified clause w.r.t. a minimum is flipped, then this clause is satisfied and at least one previously satisfied clause becomes unsatisfied. That means that this new unsatisfied clause was once-satisfied. Accordingly, the initial falsified clause was critical. \square

On Probing and Multi-Threading in PLATYPUS

J. Gressmann¹ and T. Janhunen² and R. Mercer³ and T. Schaub^{1,4} and S. Thiele¹ and R. Tichy¹

Abstract. The PLATYPUS approach offers a generic platform for distributed answer set solving, accommodating a variety of different architectures for distributing the search for answer sets across different processes and different search modes for modifying search behaviour. We describe two major extensions of PLATYPUS. First, we present its *probing* mode which provides a controlled non-linear traversal of the search space. Second, we present its new *multi-threading* architecture allowing for intra-process distribution. Both contributions are underpinned by experimental results illustrating their computational impact.

1 INTRODUCTION

The success of Answer Set Programming (ASP) has been greatly enhanced by the availability of highly efficient ASP-solvers [8, 11]. But, more complex applications are requiring computationally more powerful devices. Distributing parts of the search space among cooperating sequential solvers performing independent searches can provide increased computational power. We have proposed a generic approach to distributed answer set solving, called PLATYPUS [5].

The PLATYPUS approach differs from other pioneering work in distributed answer set solving [3, 10], by accommodating in a single design a variety of different architectures for distributing the search for answer sets over different processes. The resulting platform, *platypus*, allows one to exploit the increased computational power of clustered and/or multi-processor machines via different types of inter- and intra-process distribution techniques like MPI [7], Unix' fork mechanism, and (as discussed in the sequel) multi-threading. In addition, the generic approach permits a flexible instantiation of all parts of the design.

More precisely, the PLATYPUS design incorporates two distinguishing features: First, it modularises (and is thus independent of) the propagation engine (currently exemplified by *smodels*' and *nomore++*' expansion procedures). Second, the search space is represented explicitly. This representation allows a flexible distribution scheme to be incorporated, thereby accommodating different distribution policies and architectures. The two particular contributions discussed in this paper take advantage of these two aspects of the generic design philosophy. The first extension to PLATYPUS, *probing*, refines the encapsulated module for propagation. Probing is akin to restart in the SAT solving framework [4]. The introduction of probing demonstrates one aspect of the flexibility in our PLATYPUS design: by having a modularised generic design, we can easily specify parts of the generic design to give different computational properties to the *platypus* system. Our second improvement to

platypus is the integration of multi-threading into our software package [9]. Multi-threading expands the implemented architectural options for delegating the search space and adds several new features to *platypus*: (1) the single- and multi-threaded versions can take advantage of new hardware innovations such as multi-core processors, as well as primitives to implement lock-free data structures, (2) a hybrid architecture which allows the mixing of inter- and intra-process distribution, and (3) the intra-process distribution provides a lighter parallelisation mechanism than forking.

We highlight our two contributions, *probing* and *multi-threading*, by focusing on the appropriate aspects of the abstract PLATYPUS algorithm reproduced from [5] below. As well, their computational impact is exposed in data provided by a series of experiments.

2 THE PLATYPUS APPROACH

In ASP, a logic program Π is associated with a set of *answer sets*, $AS(\Pi)$, which are distinguished models of the rules in Π . We do not elaborate, but refer the reader to [2] for a formal introduction to ASP. For computing answer sets, we rely on *partial assignments*, mapping atoms in an alphabet \mathcal{A} onto true, false, or undefined. We represent such assignments as pairs (X, Y) of sets of atoms, in which X contains all true atoms and Y all false ones. In general, a partial assignment (X, Y) aims at capturing a subset of the answer sets of Π , viz. $AS_{(X,Y)}(\Pi) = \{Z \in AS(\Pi) \mid X \subseteq Z, Z \cap Y = \emptyset\}$.

To begin, we recapitulate the major features of the PLATYPUS approach [5]. To enable a distributed search for answer sets, the search space is decomposed by means of partial assignments. This method works because partial assignments that differ with respect to defined atoms represent different parts of the search space. To this end, Al-

Algorithm 1: PLATYPUS

Global : A logic program Π over alphabet \mathcal{A} .
Input : A nonempty set S of partial assignments.
Output : Print a subset of the answer sets of Π .

```

repeat
1    $(X, Y) \leftarrow \text{CHOOSE}(S)$ 
2    $S \leftarrow S \setminus \{(X, Y)\}$ 
3    $(X', Y') \leftarrow \text{EXPAND}((X, Y))$ 
4   if  $X' \cap Y' = \emptyset$  then
5     if  $X' \cup Y' = \mathcal{A}$  then print  $X'$  else
6        $A \leftarrow \text{CHOOSE}(\mathcal{A} \setminus (X' \cup Y'))$ 
7        $S \leftarrow S \cup \{ (X' \cup \{A\}, Y'), (X', Y' \cup \{A\}) \}$ 
8        $S \leftarrow \text{DELEGATE}(S)$ 
until  $S = \emptyset$ 

```

gorithm 1 is based on an explicit representation of the search space in terms of a set S of partial assignments, on which it iterates until S becomes empty. The algorithm relies on the omnipresence of a

¹ Universität Potsdam, Postfach 900327, D-14439 Potsdam, Germany.

² Helsinki University of Technology, P.O. Box 5400, FI-02015 TKK, Finland.

³ University of Western Ontario, London, Ontario, Canada N6A 5B7.

⁴ Affiliated with Simon Fraser University, Burnaby, Canada.

logic program Π and its alphabet \mathcal{A} as global parameters. Communication between PLATYPUS instances is limited to delegating partial assignments as representatives of parts of the search space. The set of partial assignments provided in the input variable S delineates the search space given to a specific instance of PLATYPUS. Although this explicit representation offers an extremely flexible access to the search space, it must be handled with care since it grows exponentially in the worst case. Without Line 8, Algorithm 1 computes all answer sets in $\bigcup_{(X,Y) \in S} AS_{(X,Y)}(\Pi)$. With Line 8 each PLATYPUS instance generates a subset of the answer sets. CHOOSE and DELEGATE are in principle non-deterministic selection functions: CHOOSE yields a single element, DELEGATE communicates a subset of S to a PLATYPUS instance and returns a subset of S . Clearly, depending on what these subsets are, this algorithm is subject to incomplete and redundant search behaviours. The EXPAND function hosts the deterministic part of Algorithm 1. This function is meant to be implemented with an off-the-shelf ASP-expander that is used as a black-box providing both sufficiently strong as well as efficient propagation operations. See [5] for details.

We now turn to specific design issues beyond the generic description of Algorithm 1. To reduce the size of partial assignments and thus that of passed messages, we follow [10] in representing partial assignments only by atoms⁵ whose truth values were assigned by choice operations (cf. atom A in Lines 6 and 7). Given assignment (X, Y) with its subsets $X_c \subseteq X$ and $Y_c \subseteq Y$ of atoms assigned by a choice operation, we have $(X, Y) = \text{EXPAND}((X_c, Y_c))$. Consequently, the expansion of assignment (X, Y) to (X', Y') in Line 3 does not affect the representation of the search space in S .⁶ Furthermore, the design includes the option of using a choice proposed by the EXPAND component for implementing Line 6. Additionally, the currently used expanders, smodels and nomore++, also supply a *polarity*, indicating a preference for assigning true or false.

Each `platypus` process has an explicit representation of its (part of the) search space in its variable S . This set of partial assignments is implemented as a tree. Whenever more convenient, we describe S in terms of a set of assignments or a search tree and its branches. In contrast to stack-based ASP-solvers, like smodels or nomore++, whose search space contains a single branch at a time, this tree normally contains several independent branches. The *active* partial assignment (or branch) selected in Line 1, is the one being currently treated by the expander. The state of the expander is characterised by the contents of its stack, which corresponds to the active branch in the search tree. While the stack contains the full assignment (X, Y) , the search tree's active branch only contains the pair of subsets (X_c, Y_c) .

3 PROBING

The explicit representation of the (partial) search space, although originally devised to enable the use of a variety of strategies for delegating parts of the search space in the distributed setting, appears to be beneficial in some sequential contexts, as well. Of particular interest, when looking for a single answer set, is limiting fruitless searches in parts of the search tree that are sparsely populated with answer sets. In such cases, it seems advantageous to leave a putatively sparsely populated part and continue at another location in the search space. In `platypus`, this decision is governed by two command line options, $\#c$ and $\#j$. A part of the search is regarded as fruitless, whenever the number of *conflicts* (as encountered in Line 4)

⁵ Assignments are not restricted to atoms, as used when using nomore++.

⁶ Accordingly, the tests in Lines 4 and 5 must be handled with care; see [5].

exceeds the value of $\#c$. The corresponding conflict counter⁷ c is incremented each time a conflict is detected in Line 4. The counter c is reset to zero whenever an answer set is found in Line 5 or the active branch in S is switched (and thus the expander is reinitialised; see Algorithm 2). The number of *jumps* in the search space is limited by $\#j$; each jump changes the active branch in the search space. We use a *binary exponential back-off* (cf. [12]) scheme to heed unsuccessful jumps. The idea is as follows. At first, probing initiates a jump in the search space whenever the initial conflict limit $\#c$ is reached. If no solution is found after $\#j$ jumps, then the problem appears to be harder than expected. In this case, the permissible number of conflicts $\#c$ is doubled and the allowed number of jumps $\#j$ is halved. The former is done to prolong systematic search, the latter to reduce gradually to zero the number of jumps in the search space. We refer to this treatment of the search space as *probing*. Probing is made precise in Algorithm 2, which is a refinement of the CHOOSE operation in Line 1 of Algorithm 1. Note that probing continues until the pa-

Algorithm 2: CHOOSE (in Line 1 of Algo. 1) in *probing mode*.

Global : Positive integers $\#c, \#j$, supplied via command line.
 Integers c, j , initially $c = 0$ and $j = \#j$.
 Selection policy \mathcal{P} , supplied via command line.
Input : A set S of assignments with active assignment $b \in S$.
Output: A partial assignment.
// Counter c is incremented by one in Line 4 of Algorithm 1.
if ($c \leq \#c$) or ($\#j = 0$) **then return** b ; // no jumping
else // jumping
 $c \leftarrow 0$
 $j \leftarrow j - 1$
 if ($j = 0$) **then**
 $\#c \leftarrow (\#c \times 2)$
 $\#j \leftarrow (\#j \text{ div } 2)$
 $j \leftarrow \#j$
 let $b' \leftarrow \text{SELECT}(\mathcal{P}, S)$ **in**
 ⟨make b' the active partial assignment in S ⟩
 return b'

rameter $\#j$ becomes zero. When probing stops, search proceeds in the usual depth-first manner by considering only one branch at a time by means of the expander's stack. Clearly, this is also the case during the phases when the conflict limit has not been reached ($c \leq \#c$).

At the level of implementation, the expander must be reinitialised whenever the active branch of the search space changes. Reinitialisation is unnecessary when extending the active branch by the choice (obtained in Line 6) in Line 7 of Algorithm 1 or when backtracking is possible in case a conflict occurs or an answer set is obtained. In the first case, the expander's choice (that is, an atom with a truth value) is simply pushed on top of the expander's stack (and marked as a possible backtracking point). At the same time, the active branch in S is extended by the choice and a copy of the active branch, extended by the complementary choice, is added to S . See [6] for details.

In the case that a conflict occurs or an answer set is obtained, the active branch in S is replaced by the branch corresponding to the expander's stack after backtracking. If it exists, this is the largest branch in S that equals a subbranch of the active branch after switching the truth value of its leaf element. If backtracking is impossible, the active branch is chosen by means of the given policy \mathcal{P} (at present, a largest, a smallest, or a random assignment). If this, too, is impossi-

⁷ Each thread has its own conflict and jump counters.

ble, S must be empty and the PLATYPUS instance terminates.

The policy-driven selection of a branch, expressed in Algorithm 2 by $\text{SELECT}(\mathcal{P}, S)$, is governed by another command line option⁸ $\#n$ and works in two steps. First, among all branches in S , the $\#n$ best ones, $b_1, \dots, b_{\#n}$, are identified according to policy \mathcal{P} . To be precise, let p be a mapping of branches to ordinal values, used by \mathcal{P} for evaluating branches. For every $b \in \{b_1, \dots, b_{\#n}\}$ and $b' \in S \setminus \{b_1, \dots, b_{\#n}\}$, we then have that $p(b) \leq p(b')$. Then, a branch b is randomly selected from $\{b_1, \dots, b_{\#n}\}$. This random selection from the best $\#n$ branches counteracts the effect of a rigid policy by arbitrarily choosing some close alternatives.

To see that probing guarantees completeness, it is sufficient to see that no partial assignment is ever eliminated from the search space. Also, when probing, the number of different branches in the search space S cannot exceed twice the number of initially permitted jumps, viz. $2 \times \#j$. For instance, if the command line option sets $\#j$ to 13, we may develop at most $13 + 6 + 3 + 1$ different branches in S , which is bound by 2×13 . Thereby, a branch is considered as different if it is not obtainable from another's subbranch by switching the assigned truth value of a single atom (i.e. if it is not a backtracking point).

4 THREAD ARCHITECTURE

This section details the multi-threaded architecture extension to the *platypus* platform which adds the capacity to do intra-process distribution delegation to the existing inter-process capabilities, which are optionally realised via Unix' forking mechanism or MPI [7] (described in [5]). This richer architecture now permits hybrid delegation methods, for instance, delegating *platypus* via MPI on a cluster of multi-processor machines, with delegation among the multi-processors of each machine accomplished with multi-threading.

The architecture is split into more or less two parts: the *core* and the *distribution* components. A core encapsulates the search for answer sets, and the *DELEGATE* function is encapsulated in a distribution component. The core and distribution components have well-defined interfaces that localise the communication between the components. This design allows us to incorporate, for instance, single- and multi-threaded cores, as well as inter-process distribution schemes, like MPI and forking, with ease.

Each *platypus* process hosts an instance of the core, the core object, which cooperates with one instance of the distribution component, the distribution object. Communication is directed from core to distribution objects and is initiated by the core object. During execution the major flow of control lies with the core objects.

The multi-threaded core works according to the master/slave principle. The master coordinates a number of slave threads. Each slave thread executes the PLATYPUS algorithm on its thread-local search space. The master thread handles communication (through the distribution object) with other *platypus* processes on behalf of the slave threads. Communication between the master thread and its slave threads is based on counters and queues: Events of interest (e.g. statistics, answer sets, etc.) are communicated by the slaves to the master by incrementing the appropriate counter or adding to the respective queue. The master thread periodically polls the counters and queues for any change. The search ends (followed by termination of the *platypus* program) if there is agreement among the distribution objects that either all participating processes are in need of work (indicating all the work is done) or the requested number of answer sets is computed. In the core, the *idle thread counter* of the master thread serves two purposes: It indicates the number of idle slave

threads in the core object, and it shows the number of partial assignments in the *thread delegation queue* of the master thread. Slave threads share their search space automatically among themselves as long as one thread has some work left. A slave thread running out of work (reaching an empty search space S) checks the availability of work via the idle thread counter and if possible removes a partial assignment from the thread delegation queue. Otherwise, it waits until new work is assigned to it. Another slave thread can become aware of the existence of an idle thread by noting that the idle thread counter exceeds zero during one of its periodic checks. If this is the case, it splits off a part of its local search space according to a distribution policy, puts the partial assignment that represents the subspace into the thread delegation queue, and decrements the idle thread counter. As this may happen simultaneously in several working slave threads, more partial assignments can end up in the thread delegation queue than there exist idle slaves. These extras are used subsequently by idle threads.

When all slave threads are idle (i.e. the idle thread counter equals the number of slave threads) the master thread initiates communication via the distribution object to acquire more work from other PLATYPUS processes. To this end, the master thread periodically queries the associated distribution object for work until it either gets some work or is requested to terminate. Once work is available, the master thread adds it to the thread delegation queue, decrements the idle thread counter,⁹ and wakes up a slave thread. The awoken slave thread will find the branch there, take it out, and start working again. From there on, the core enters its normal thread-to-thread mode of work sharing. Conversely, when a *platypus* process receives notification that another process has run out of work, it attempts to delegate a piece of its search space. To this end, it sets the *other-process-needs-work* flag of the master thread in its core object. All slave threads noticing this flag clear the flag and delegate a piece of their search space according to the delegation policy by adding it to the *remote delegation queue*. The master thread takes one branch out of the queue and forwards it to the requesting *platypus* process (via the distribution object). Because of the multi-threaded nature any number of threads can end up delegating. Items left in the remote delegation queue are used by the master thread to fulfil subsequent requests for work by other *platypus* processes or work requests by its slave threads. The conceptual difference between the thread delegation and the remote delegation queues is that the former handle intra-core delegations, while the latter deal with extra-core delegation, although non-delegated work can return to the core. This is reflected by the fact that master and slave threads are allowed to insert partial assignments into the thread delegation queue, whereas only slave threads remove items from this queue. In contrast, only the master thread is allowed to eliminate items from the remote delegation queue, while insertions are performed only by slave threads.

An important aspect of the multi-threaded core implementation is the use of *lock-free data structures* for synchronising communication among master and slave threads. This is detailed in [6].

5 EXPERIMENTAL RESULTS

The following experiments aim to provide some indication of the computational value of probing and multi-threading. All experiments were conducted with some fixed parameters: (i) smodels (2.28) was used as propagation engine and for delivering the (signed) choice in Line 6 of Algorithm 1, (ii) the choice in Line 1 of Algorithm 1 was

⁸ Option $\#n$ can be zero, indicating the use of all branches.

⁹ The inserting thread is responsible for decrementing the idle thread counter.

<i>clumpy</i>	<i>sm</i>	<i>st</i>	10,32	10,64	10,128	10,256	10,512	50,32	50,64	50,128	50,256	50,512	100,32	100,64	100,128	100,256	100,512	200,32	200,64	200,128	200,256	200,512
06,06,02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	
06,06,03	0.10	0.10	0.05	0.05	0.05	0.05	0.07	0.07	0.07	0.07	0.07	0.11	0.11	0.11	0.11	0.17	0.16	0.16	0.16	0.16	0.16	
06,06,04	0.61	0.63	0.08	0.08	0.08	0.08	0.14	0.14	0.14	0.14	0.14	0.24	0.24	0.24	0.24	0.34	0.34	0.34	0.34	0.34	0.34	
06,06,05	6.30	6.61	1.24	1.79	0.95	0.84	0.84	0.78	0.66	0.66	0.66	0.96	0.96	0.96	0.96	2.29	2.14	2.14	2.14	2.14	2.14	
06,06,06	0.38	0.39	0.05	0.05	0.05	0.05	0.04	0.04	0.04	0.04	0.04	0.06	0.06	0.06	0.06	0.10	0.10	0.10	0.10	0.10	0.10	
06,06,07	0.04	0.03	0.14	0.14	0.14	0.14	0.14	0.14	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.03	0.03	0.03	0.03	0.03	0.03	
06,06,08	0.08	0.08	0.01	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.03	0.03	0.03	0.03	0.03	
06,06,09	11.3	11.8	0.47	0.52	0.62	0.62	0.62	1.07	1.01	1.01	1.01	2.23	2.06	2.06	2.06	3.06	3.46	3.46	3.46	3.46	3.46	
06,06,10	0.06	0.05	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.03	0.03	0.03	0.05	0.05	0.05	0.05	0.05	0.05	0.05	
07,07,01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	
07,07,02	0.05	0.04	0.61	0.74	0.71	0.71	0.71	1.76	1.45	1.45	1.45	2.01	2.92	2.91	2.91	2.90	0.04	0.04	0.04	0.04	0.04	
07,07,03	8.98	9.60	18.7	9.56	14.5	3.75	3.26	4.79	4.72	16.9	6.11	6.05	5.02	33.8	18.4	9.71	10.3	23.3	9.75	22.1	14.5	
07,07,04	1.37	1.38	0.98	2.05	2.01	3.49	3.38	1.57	1.79	1.54	1.54	1.53	2.87	2.19	2.19	2.20	2.19	2.76	3.30	3.30	3.28	
07,07,05	0.03	0.02	0.04	0.04	0.04	0.04	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	
07,07,06	0.38	0.38	0.41	0.38	0.38	0.38	0.38	0.61	0.61	0.61	0.61	0.61	0.69	0.69	0.69	0.86	0.86	0.86	0.86	0.86	0.86	
07,07,07	0.04	0.03	0.08	0.08	0.08	0.08	0.08	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	
07,07,08	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.14	0.14	0.14	0.14	0.14	0.14	
07,07,09	0.40	0.40	0.08	0.08	0.08	0.08	0.08	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.55	0.55	0.55	0.55	0.55	0.55	
07,07,10	124.5	126.4	15.8	6.32	2.17	1.96	1.97	31.7	13.4	6.01	5.27	5.27	59.3	72.0	9.49	8.74	8.74	18.8	21.5	20.4	14.1	14.1
08,08,01	5.07	1.64	2.44	4.68	5.23	22.5	2.84	3.21	3.22	3.20	10.9	4.81	4.76	4.72	4.68	45.1	15.4	10.3	10.2	10.0	10.0	
08,08,02	7.04	11.1	2.42	2.44	2.43	8.01	6.22	5.61	6.64	6.61	23.0	12.0	9.74	9.05	8.98	44.0	15.5	13.7	13.8	13.7	13.7	
08,08,03	14.8	9.39	13.1	5.31	5.52	61.9	84.9	7.57	14.0	13.1	105.8	51.8	9.17	8.71	8.66	32.8	205.8	15.9	15.3	15.3	15.3	
08,08,05	36.7	37.0	231.2	16.1	33.6	43.6	176.6	24.1	36.1	53.5	96.5	48.3	29.2	47.7	84.1	129.2	70.0	39.4	87.3	189	240	
08,08,06	8.15	8.22	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.10	0.10	0.16	0.17	0.17	0.17	0.26	0.26	0.26	0.26	0.26	0.26	
08,08,07	4.17	4.10	0.44	0.44	0.44	0.44	0.43	1.23	1.24	1.23	1.23	0.48	0.48	0.48	0.48	0.47	0.89	0.90	0.90	0.90	0.89	
08,08,08	0.85	71.6	14.5	6.33	13.5	2.16	1.73	1.73	1.72	1.72	3.69	2.77	2.77	2.77	2.76	6.40	4.76	4.76	4.76	4.76	4.75	
08,08,09	1.29	0.87	0.88	0.88	0.87	1.07	1.08	1.08	1.08	1.07	2.03	2.03	2.03	2.03	2.02	3.02	3.04	3.03	3.03	3.02	3.02	
08,08,10	1.66	1.67	17.3	11.5	4.24	4.37	4.02	1.87	2.24	2.24	2.23	4.93	2.72	2.72	2.72	5.97	7.41	7.41	7.40	7.37	7.37	
09,09,01	24.9		0.34	0.34	0.34	0.34	0.10	0.10	0.10	0.10	0.10	0.11	0.11	0.11	0.11	0.12	0.12	0.12	0.12	0.12	0.12	
09,09,02			1.66	1.82	2.84	2.64	2.63	0.85	0.85	0.85	0.84	1.48	1.49	1.49	1.49	1.48	2.31	2.32	2.33	2.32	2.31	
09,09,03			13.3	4.24	7.33	74.3	0.82	0.82	0.82	0.82	0.82	1.67	1.68	1.68	1.68	1.68	2.51	2.52	2.52	2.51	2.51	
09,09,04			143.8			50.9					81.6					95.7						
09,09,05			2.60	2.08	2.66	2.66	4.03	3.98	4.68	4.68	4.67	3.96	4.80	4.81	4.80	4.79	6.49	6.32	6.31	6.33	6.31	
09,09,06			4.00	2.59	159.6	6.40	5.89	11.5	8.62	5.51	5.51	5.50	7.35	21.5	6.45	6.46	6.44	12.8	20.1	17.4	17.4	
09,09,07			0.75	28.4	3.23	3.01	2.16	2.03	2.04	2.03	2.03	3.05	3.07	3.06	3.05	6.70	5.95	5.95	5.95	5.90	5.90	
09,09,09			0.73	0.71	0.71	0.71	1.95	2.40	2.40	2.40	2.39	3.91	3.50	3.51	3.50	3.48	12.5	9.68	9.67	9.69	9.63	

Table 1. Experimental results for *probing* (with the single-threaded core).

fixed to the policy selecting assignments with the largest number of unassigned atoms, (iii) all such selections were done in a deterministic way by setting command-line option $\#n$ to 1 (cf. Section 2). All tests were done with *platypus* version 0.2.2 [9]. They reflect average times of 5 runs for finding the first or all answer sets, resp., of the considered instance. Timing excludes parsing and printing. The data was obtained on a quad processor (4 Opteron 2.2GHz processors, 8 GB shared RAM) under Linux.

For illustrating the advantage of probing, we have chosen the search for one Hamiltonian cycle in *clumpy graphs*, proposed in [13] as a problem set being problematic for systematic backtracking. These benchmarks are available at [9]. Table 1 contrasts different settings for numbers of conflicts $\#c$ (10, 50, 100, 200) and jumps $\#j$ (32, 64, 128, 256, 512), resp., running the single-threaded core. For comparison, we also provide the corresponding *smodels* times¹⁰ and the ones for single-threaded *platypus* without probing in the columns labelled *sm* and *st*. The remaining columns are labelled with the command line options used, viz. $\#c, \#j$. A blank entry represents a timeout after 240 seconds. First of all, we notice that the systems using standard depth first-search are unable to solve 12 instances within the given time limit, whereas when using probing, apart for a few exceptions, all instances are solved. We see that *platypus* without probing does best 8 times, as indicated in bold-face, and worst 24 times, whereas *smodels* does best 2 times¹¹ and worst 24 times. Compared to each specific probing configuration, *platypus* without probing performs better among 9 to 15 (*smodels*, 6 to 8) times out of 38. In fact, there seems to be no clear pattern indicating a best probing configuration. However, looking at the lower part of Table 1, we observe that *platypus* without probing (*smodels*) times out 12 times, while probing still gives a

¹⁰ These times are only indicative since they include printing one answer set.
¹¹ The six cases differ by only 0.01sec which is due to slightly different timing methods (see Footnote 10).

solution under all but three configurations. In all, we see that probing allows for a significant speed-up for finding the first answer set. This is particularly valuable whenever answer sets are hard to find with a systematic backtracking procedure, as witnessed by the entries in the lower part of Table 1. However, probing has generally no positive effect when computing all answer sets. Also, on more common benchmarks (cf. [1]) probing rarely kicks in since the conflict counter is earlier reset to zero whenever an answer set is found.

The computational impact of probing is even more significant when using multi-threading,¹² where further speed-ups are observed on 20 benchmarks, most of which are among the more substantial ones in the lower part of Table 1. The most substantial one is observed on clumpy graph 09,09,04 which is solved in 4.66 and 4.26 seconds, resp., when setting $\#c, \#j$ to 10,512 and using 3 and 4 slave threads, resp. Interestingly, even the multi-threaded variant *without* probing cannot solve the last seven benchmarks within the time limit, except for clumpy 09,09,07, which *platypus* with 4 slave threads is able to solve in 13.8 seconds. This illustrates that probing and multi-threading are two complementary techniques that can be used for accelerating the performance of standard ASP-solvers. A way to tackle benchmarks that are even beyond the reach of probing with multi-threading is to use randomisation via command-line option $\#n$.

Table 2 displays the effect of multi-threading, when computing all answer sets. For consistency, we have taken a subset of the asparagus benchmarks [1] in [5], used when evaluating the speed-ups obtained with the (initial) forking and MPI variant of *platypus*. Comparing the sum of the average times, the current *platypus* variant running multi-threading is 2.64 times faster than its predecessor using forking, as reported in [5].¹³ In more detail, the columns reflect the times of *platypus* run with the multi-threaded core restricted

¹² All tests on multi-threading with and without probing are provided at [9].

¹³ The forking tests in [5] were also run on the same machine.

problem	mt #1	mt #2	mt #3	mt #4
color-5-10	1.53	0.84	0.62	0.53
color-5-15	60.9	31.1	20.5	15.7
ham_comp_8	3.66	1.99	1.38	1.10
ham_comp_9	85.2	43.6	29.0	22.5
pigeon-7-8	1.38	0.73	0.57	0.48
pigeon-7-9	4.22	2.19	1.46	1.17
pigeon-7-10	13.2	6.31	4.12	3.08
pigeon-7-11	36.5	16.3	10.6	7.94
pigeon-7-12	88.2	39.9	25.8	19.0
pigeon-8-9	11.6	5.77	3.80	2.84
pigeon-8-10	48.3	22.3	14.2	10.4
pigeon-9-10	128.4	61.8	39.5	29.4
schur-14-4	1.00	0.63	0.47	0.42
schur-15-4	2.38	1.30	0.91	0.73
schur-16-4	4.04	2.14	1.41	1.11
schur-17-4	9.13	4.58	3.04	2.28
schur-18-4	16.7	8.34	5.31	3.92
schur-19-4	39.3	18.1	11.5	8.28
schur-20-4	44.1	21.9	13.8	10.1
schur-11-5	0.56	0.37	0.32	0.32
schur-12-5	1.49	0.83	0.63	0.54
schur-13-5	5.69	2.90	1.97	1.51
schur-14-5	18.6	9.05	6.00	4.42

Table 2. Experimental results on *multi-threading*.

to 1, 2, 3, and 4 slave threads (probing disabled). When looking at each benchmark, the experiments show a qualitatively consistent 2-, 3-, and 4-times speed-up when doubling, tripling, and quadrupling the number of processors, with only minor exceptions. For instance, the smallest speed-up is observed on *schur-11-5* (1.52, 1.73, 1.75); among the highest speed-ups, we find *schur-19-4* (2.17, 3.43, 4.75) and *pigeon-7-11* (2.24, 3.43, 4.6). The average speed-ups observed on this set of benchmarks is 1.96, 2.89, and 3.75. If we weight the average speed-ups with the respective average running times, we obtain even a slightly super-linear speed-up: 2.07, 3.18, 4.24. Such super-linear speed-ups are observed primarily on time-demanding benchmarks and, although less significant, have also been observed in [5] when forking (which makes us ascribe them to caching effects and/or shared memory). In all, we observe that the more substantial the benchmark, the more clear-cut the speed-up.

Given that the experiments were run on a quad processor, it is worth noting that we observe no drop in performance when increasing the number of slave threads from 3 to 4, despite having a fifth (master) thread. Finally, we note that the multi-threaded core, when restricted to a single slave thread, loses on average only 2% performance compared to the single-threaded version.

6 DISCUSSION

At the heart of the PLATYPUS design is its generality and modularity. These two features allow a great deal of flexibility in any instantiation of the algorithm, making it unique among related approaches. Up to now, this flexibility was witnessed by the possibility to use different off-the-shelf solvers, different process-oriented distribution mechanisms, and a variety of choice policies. In this paper we have presented two significant configurable enhancements to platypus.

First, we have described its probing mode, relying on an explicit yet restricted representation of the search space. This provides us with a global view of the search space and allows us to have different threads working on different subspaces. Although probing does not primarily aim at a sequential setting, we have experimentally demon-

strated its computational value on a specific class of benchmarks, which is problematic for standard ASP-solvers. Unlike restart strategies in SAT, which usually draw on learnt information [4], probing keeps previously abandoned parts of the search space, so that they can be revisited subsequently. Probing offers a non-linear¹⁴ exploration of the search space that can be randomised while remaining complete, a search strategy that no other native ASP-solver offers.

Second, we have presented *platypus*' multi-threaded architecture. Multi-threading complements the previous process-oriented distribution schemes of *platypus* by providing further intra-process distribution capacities. This is of great practical value since it allows us to take advantage of recent hardware developments, offering multi-core processors. In a hybrid setting, consisting of clusters of such machines, we may use multi-threading for distribution on the multi-core processors, while distribution among different workstations is done with previously established distribution techniques in *platypus*, like MPI. Furthermore, the modular implementation of the *core* and *distribution* component allow for easy modifications in view of new distribution concepts, like grid computing, for instance. The *platypus* platform is freely available on the web [9].

For more details and related work, we refer the reader to [6].

ACKNOWLEDGMENTS. Research at Potsdam was supported by DFG (SCHA 550/6-4), and at U.W.O. by NSERC (Canada) and SHARCNET. We are grateful to C. Anger, M. Brain, M. Gebser, B. Kaufmann, and the referees for many helpful suggestions.

REFERENCES

- [1] <http://asparagus.cs.uni-potsdam.de>.
- [2] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.
- [3] R. Finkel, V. Marek, N. Moore, and M. Truszczynski, ‘Computing stable models in parallel’, in *Proc. of AAAI Spring Symposium on Answer Set Programming (ASP’01)*, eds., A. Provetti and T. Son, pp. 72–75. AAAI/MIT Press, (2001).
- [4] C. Gomes, B. Selman, and H. Kautz, ‘Boosting combinatorial search through randomization’, in *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI’98)*, pp. 431–437. AAAI Press, (1998).
- [5] J. Gressmann, T. Janhunen, R. Mercer, T. Schaub, S. Thiele, and R. Tichy, ‘Platypus: A platform for distributed answer set solving’, in *Proc. of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’05)*, eds., C. Baral, G. Greco, N. Leone, and G. Terracina, pp. 227–239. Springer-Verlag, (2005).
- [6] J. Gressmann, T. Janhunen, R. Mercer, T. Schaub, S. Thiele, and R. Tichy, ‘On probing and multi-threading in platypus’, in *Proc. of the Eleventh International Workshop on Nonmonotonic Reasoning*, eds., J. Dix and A. Hunter, (2006). To appear.
- [7] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*, The MIT Press, 1999.
- [8] N. Leone, W. Faber, G. Pfeifer, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri, and F. Scarcello, ‘The DLV system for knowledge representation and reasoning’, *ACM TOCL*, (2006). To appear.
- [9] <http://www.cs.uni-potsdam.de/platypus>.
- [10] E. Pontelli, M. Balduccini, and F. Bermudez, ‘Non-monotonic reasoning on beowulf platforms’, in *Proc. of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL’03)*, eds., V. Dahl and P. Wadler, pp. 37–57. Springer-Verlag, (2003).
- [11] P. Simons, I. Niemelä, and T. Soininen, ‘Extending and implementing the stable model semantics’, *Art. Intell.*, **138**(1-2), 181–234, (2002).
- [12] A. S. Tanenbaum, *Modern Operating Systems*, Prentice Hall, 2001.
- [13] J. Ward and J. Schlipf, ‘Answer set programming with clause learning’, in *Proc. of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’04)*, eds., V. Lifschitz and I. Niemelä, pp. 302–313. Springer-Verlag, (2004).

¹⁴ That is, the traversal of the search space does not follow a given strategy like depth-first search.

Elaborating domain descriptions

Andreas Herzig and Laurent Perrussel and Ivan Varzinczak¹

Abstract. In this work we address the problem of *elaborating* domain descriptions (alias action theories), in particular those that are expressed in dynamic logic. We define a general method based on contraction of formulas in a version of propositional dynamic logic with a solution to the frame problem. We present the semantics of our theory change and define syntactical operators for contracting a domain description. We establish soundness and completeness of the operators w.r.t. the semantics for descriptions that satisfy a principle of modularity that we have defined in previous work.

1 INTRODUCTION

Suppose a situation where an agent has always believed that if the light switch is up, then the room is light. Suppose now that someday, she observes that even if the switch is up, the light is off. In such a case, the agent must change her beliefs about the relation between the propositions “the switch is up” and “the light is on”. This is an example of changing propositional belief bases and is largely addressed in the literature about belief change [10] and update [24].

Next, let our agent believe that whenever the switch is down, after toggling it, the room is light. This means that if the light is off, in every state of the world that follows the execution of toggling the switch, the room is lit up. Then, during a blackout, the agent toggles the switch and surprisingly the room is still dark.

Imagine now that the agent never worried about the relation between toggling the switch and the material it is made of, in the sense that she ever believed that just toggling the switch does not break it. Nevertheless, in a stressful day, she toggles the switch and then observes that she had broken it.

Completing the wayside cross our agent experiments in discovering the world’s behavior, suppose she believed that it is always possible to toggle the switch, given some conditions e.g. being close enough to it, having a free hand, the switch is not broken, etc. However, in an April fool’s day, she discovers that someone has glued the switch and, consequently, it is no longer possible to toggle it.

The last three examples illustrate situations where changing the beliefs about the behavior of the action of toggling the switch is mandatory. In the first one, toggling the switch, once believed to be deterministic, has now to be seen as nondeterministic, or alternatively to have a different outcome in a specific context (e.g. if the power station is overloaded). In the second example, toggling the switch is known to have side-effects (ramifications) one was not aware of. In the last example, the executability of the action under concern is questioned in the light of new information showing a context that was not known to preclude its execution. Carrying out such modifications is what we here call *elaborating* a domain description, which has to do with the principle of *elaboration tolerance* [28].

¹ The authors are with the Institut de Recherche en Informatique de Toulouse (IRIT), Toulouse, France. e-mail: {herzig, perrusse, ivan}@irit.fr

Such cases of theory change are very important when one deals with logical descriptions of dynamic domains: it may always happen that one discovers that an action actually has a behavior that is different from that one has always believed it had.

Up to now, theory change has been studied mainly for knowledge bases in classical logics, both in terms of revision and update. Only in a few recent works it has been considered in the realm of modal logics, viz. in epistemic logic [12] and in action languages [7]. Recently, several works [31, 21] have investigated revision of beliefs about facts of the world. In our examples, this would concern e.g. the current status of the switch: the agent believes it is up, but is wrong about this and might subsequently be forced to revise her beliefs about the current state of affairs. Such revision operations do not modify the agent’s beliefs about the action laws. In opposition to that, here we are interested exactly in such modifications. The aim of this paper is to make a step toward that issue and propose a framework that deals with the contraction of action theories.

Propositional dynamic logic (PDL [13]), has been extensively used in reasoning about actions in the last years [2, 36, 8]. It has shown to be a viable alternative to situation calculus approaches because of its simplicity and existence of proof procedures for it. In this work we investigate the elaboration of domain descriptions encoded in a simplified version of such a logical formalism, viz. the multimodal logic K_n . We show how a theory expressed in terms of static laws, effect laws and executability laws is elaborated: usually, a law has to be changed due to its generality, i.e., the law is too strong and has to be weakened. It follows that elaborating an action theory means contracting it by static, effect or executability laws, before expanding the theory with more specific laws.

2 BACKGROUND

Following the tradition in the reasoning about actions community, action theories are collections of statements of the form: “if *context*, then *effect* after every execution of *action*” (effect laws); and “if *pre-condition*, then *action executable*” (executability laws). Statements mentioning no action at all represent laws about the world (static laws). Besides that, statements of the form “if *context*, then *effect* after some execution of *action*” will be used as a causal notion to solve the frame and the ramification problems.

2.1 Logical preliminaries

Let $\mathcal{Act} = \{a_1, a_2, \dots\}$ be the set of all *atomic actions* of a given domain, an example of which is *toggle*. To each atomic action a there is associated a modal operator $[a]$. $\mathcal{Prop} = \{p_1, p_2, \dots\}$ denotes all the *propositional constants* (alias *fluents* or *atoms*). Examples of those are *light* (“the light is on”) and *up* (“the switch is up”). The set of all literals is $\mathcal{Lit} = \mathcal{Prop} \cup \{\neg p : p \in \mathcal{Prop}\}$.

\mathfrak{Fml} is the set of all classical formulas. They are denoted by small Greek letters φ, ψ, \dots . An example is $up \rightarrow light$. By $val(\varphi)$ we denote the set of valuations making φ true. We view a valuation as a maximally-consistent set of literals. For $\mathfrak{Prop} = \{light, up\}$, there are four valuations: $\{light, up\}$, $\{light, \neg up\}$, $\{\neg light, up\}$ and $\{\neg light, \neg up\}$. Given a set of formulas Σ , by $lit(\Sigma)$ we denote the set of all literals appearing in formulas of Σ .

We denote complex formulas (with modal operators) by Φ, Ψ, \dots . $\langle a \rangle$ is the dual operator of $[a]$, defined as $\langle a \rangle \Phi =_{\text{def}} \neg [a] \neg \Phi$. An example of a complex formula is $\neg up \rightarrow [toggle]up$. The semantics is that of multimodal logic K [29].

A K_n -model is a tuple $\mathcal{M} = \langle W, R \rangle$ where W is a set of valuations, and R a function mapping action constants a to accessibility relations $R_a \subseteq W \times W$. Given a K_n -model $\mathcal{M} = \langle W, R \rangle$, $\models_w^{\mathcal{M}} p$ (p is true at world w of model \mathcal{M}) if $p \in w$; $\models_w^{\mathcal{M}} [a]\Phi$ if for every w' such that wR_aw' , $\models_{w'}^{\mathcal{M}} \Phi$. Truth conditions for other connectives are as usual.

\mathcal{M} is a model of Φ (noted $\models^{\mathcal{M}} \Phi$) if for all $w \in W$, $\models_w^{\mathcal{M}} \Phi$. \mathcal{M} is a model of a set of formulas Σ (noted $\models^{\mathcal{M}} \Sigma$) if $\models^{\mathcal{M}} \Phi$ for every $\Phi \in \Sigma$. Φ is a consequence of the set of global axioms Γ in the class of all K_n -models (noted $\Gamma \vdash_{K_n} \Phi$) if for every K_n -model \mathcal{M} , $\models^{\mathcal{M}} \Gamma$ implies $\models^{\mathcal{M}} \Phi$.

2.2 Describing the behavior of actions in K_n

K_n allows for the representation of statements describing the behavior of actions. They are called *action laws*. Here we distinguish several types of them. The first kind of statement represents the *static laws*, formulas that must hold in every possible state of the world.

Definition 1 A static law is a formula $\varphi \in \mathfrak{Fml}$.

An example of a static law is $up \rightarrow light$: if the switch is up, then the light is on. $\mathcal{S} \subseteq \mathfrak{Fml}$ denotes all the static laws of a domain.

The second kind of law we consider are the *effect laws*. They are formulas relating an action to its effects, which can be conditional.

Definition 2 An effect law for action a has the form $\varphi \rightarrow [a]\psi$, where $\varphi, \psi \in \mathfrak{Fml}$.

The consequent ψ is the effect always obtained when a is executed in a state where the antecedent φ holds. \mathcal{E} denotes the set of all effect laws of a domain, an example of which is $\neg up \rightarrow [toggle]light$: whenever the switch is down, after toggling it, the room is lit up. If ψ is inconsistent, we have a special kind of effect law that we call an *inexecutability law*. For example, $broken \rightarrow [toggle]\perp$ says that *toggle* cannot be executed if the switch is broken.

Finally, we also define *executability laws*, which stipulate the context for an action to be executable. In K_n , we use $\langle a \rangle$ to express executability. $\langle a \rangle \top$ thus reads “the execution of a is possible”.

Definition 3 An executability law for action a is of the form $\varphi \rightarrow \langle a \rangle \top$, where $\varphi \in \mathfrak{Fml}$.

For instance, $\neg broken \rightarrow \langle toggle \rangle \top$ says that toggling can be executed whenever the switch is not broken. The set of all executability laws of a given domain is denoted by \mathcal{X} .

3 MODELS OF CONTRACTION

When an action theory has to be changed, the basic operation is that of *contraction*. (In belief-base update [33, 24] it has also been called *erasure*.) In this section we define its semantics.

In general we might contract by any formula Φ . Here we focus on contraction by one of the three kinds of laws. We therefore suppose that Φ is either φ , where φ is classical, or $\varphi \rightarrow [a]\psi$, or $\varphi \rightarrow \langle a \rangle \top$.

For the case of contracting static laws we resort to existing approaches to change the set of static laws. In the following, we consider any belief change operator like Forbus’ update method [9], the possible models approach [33, 34], WSS [14] or MPMA [6].

Contraction by φ corresponds to adding new possible worlds to W . Let \ominus be a given contraction operator for classical logic.

Definition 4 Let $\langle W, R \rangle$ be a K_n -model and φ a classical formula. The model resulting from contracting by φ is $\langle W, R \rangle_{\varphi}^- = \{\langle W', R' \rangle\}$ such that $W' = W \ominus val(\varphi)$.

Observe that R should, a priori, change as well, otherwise contracting a classical formula may conflict with \mathcal{X} .² For instance, if $\neg \varphi \rightarrow \langle a \rangle \top \in \mathcal{X}$ and we contract by φ , the result may make \mathcal{X} untrue. However, given the amount of information we have at hand, we think that whatever we do with R (adding or removing edges), we will always be able to find a counter-example to the intuitiveness of the operation, since it is domain dependent. For instance, adding edges for a deterministic action may render it nondeterministic. Deciding on what changes to carry out on R when contracting static laws depends on the user’s intuition, and unfortunately this information cannot be generalized and established once for all. We here opt for a priori doing nothing with R and postponing correction of executability laws.

Action theories being defined in terms of effect and executability laws, elaborating an action theory will mainly involve changes in these two sets of laws. Let us consider now both these cases.

Suppose the knowledge engineer acquires new information regarding the effect of action a . Then it means that the law under consideration is probably too strong, i.e., the expected effect may not occur and thus the law has to be weakened. Consider e.g. $\neg up \rightarrow [toggle]light$, and suppose it has to be weakened to the more specific $(\neg up \wedge \neg blackout) \rightarrow [toggle]light$.³ In order to carry out such a weakening, first the designer has to contract the set of effect laws and second to expand the resulting set with the weakened law.

Contraction by $\varphi \rightarrow [a]\psi$ amounts to adding some ‘counterexample’ arrows from φ -worlds to $\neg \psi$ -worlds. To ease such a task, we need a definition. Let $PI(\varphi)$ denote the set of prime implicants of φ . If $\varphi_1, \varphi_2 \in \mathfrak{Fml}$, $NewCons_{\varphi_1}(\varphi_2) = PI(\varphi_1 \wedge \varphi_2) \setminus PI(\varphi_1)$ computes the new consequences of φ_2 w.r.t. φ_1 : the set of strongest clauses that follow from $\varphi_1 \wedge \varphi_2$, but do not follow from φ_1 alone (cf. e.g. [20]). For example, the set of prime implicants of p_1 is just $\{p_1\}$, that of $p_1 \wedge (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee p_3 \vee p_4)$ is $\{p_1, p_2, p_3 \vee p_4\}$, hence $NewCons_{p_1}((\neg p_1 \vee p_2) \wedge (\neg p_1 \vee p_3 \vee p_4)) = \{p_2, p_3 \vee p_4\}$.

Definition 5 Let $\langle W, R \rangle$ be a K_n -model and $\varphi \rightarrow [a]\psi$ an effect law. The set of models that result from contracting by $\varphi \rightarrow [a]\psi$ is $\langle W, R \rangle_{\varphi \rightarrow [a]\psi}^- = \{\langle W, R \cup R'_a \rangle : R'_a \subseteq \{(w, w') : \models_w^{(W, R)} \varphi, \models_{w'}^{(W, R)} \neg \psi \text{ and } w' \setminus w \subseteq lit(NewCons_{\varphi}(\neg \psi))\}\}$.

In our context, $w' \setminus w \subseteq lit(NewCons_{\varphi}(\neg \psi))$ means that for all the added arrows, the new/extraneous effects of action a are limited to

² We are indebted to the anonymous referees for pointing this out to us.

³ Replacing the law by $\neg up \rightarrow [toggle](light \vee \neg light)$ looks silly.

the consequences of the static laws combined with $\neg\psi$, i.e., all the ramifications that action a can produce.

Suppose now the knowledge engineer learns new information about the executability of a . This usually occurs when some executabilities are too strong, i.e., the condition in the theory guaranteeing the executability of a is too weak and should be made more restrictive. Let e.g. $\langle \text{toggle} \rangle \top$ be the law to be contracted, and suppose it has to be weakened to the more specific $\neg\text{broken} \rightarrow \langle \text{toggle} \rangle \top$. To do that, the designer first contracts the executability laws and then expands the resulting set with the weakened law.

Contraction by $\varphi \rightarrow \langle a \rangle \top$ corresponds to removing some arrows leaving worlds where φ holds. Removing such arrows has as consequence that a is no longer always executable in context φ .

Definition 6 Let $\langle W, R \rangle$ be a K_n -model and $\varphi \rightarrow \langle a \rangle \top$ an executability law. The set of models that result from the contraction by $\varphi \rightarrow \langle a \rangle \top$ is $\langle W, R \rangle_{\varphi \rightarrow \langle a \rangle \top}^- = \{ \langle W, R' \rangle : R' = R \setminus R_a'', R_a'' \subseteq \{(w, w') : wR_a w' \text{ and } \models_w^{\langle W, R \rangle} \varphi\} \}$.

4 CONTRACTING AN ACTION THEORY

Having established the semantics of action theory contraction, we can turn to its syntactical counterpart. Nevertheless, before doing that we have to consider an important issue. As the reader might have expected, K_n alone does not solve the frame problem. For instance,

$$\left\{ \begin{array}{l} up \rightarrow light, \neg up \rightarrow [\text{toggle}]up, \\ up \rightarrow [\text{toggle}] \neg up, \langle \text{toggle} \rangle \top \end{array} \right\} \not\models_{K_n} \text{broken} \rightarrow [\text{toggle}] \text{broken}.$$

We need thus a consequence relation powerful enough to deal with the frame and ramification problems. Hence the deductive power of K_n has to be augmented to ensure that all the relevant frame axioms apply. Following the framework developed in [2], we consider metalogical information given in the form of dependence:

Definition 7 (Dependence relation [2]) A dependence relation is a binary relation $\rightsquigarrow \subseteq \mathfrak{Act} \times \mathfrak{Lit}$.

The expression $a \rightsquigarrow l$ denotes that the execution of action a may change the truth value of the literal l . On the other hand, $\langle a, l \rangle \notin \rightsquigarrow$ (written $a \not\rightsquigarrow l$) means that l can never be caused by a . In our example we have $\text{toggle} \rightsquigarrow light$ and $\text{toggle} \rightsquigarrow \neg light$, which means that action toggle may cause a change in literals $light$ and $\neg light$. We do not have $\text{toggle} \rightsquigarrow \neg broken$, for toggling the switch never repairs it.

Definition 8 A model of a dependence relation \rightsquigarrow is a K_n -model \mathcal{M} such that $\models^{\mathcal{M}} \{ \neg l \rightarrow [a] \neg l : a \not\rightsquigarrow l \}$.

We assume \rightsquigarrow is finite. Given a dependence relation \rightsquigarrow , the associated consequence relation in the set of models for \rightsquigarrow is noted $\models_{\rightsquigarrow}$. For our example we obtain

$$\left\{ \begin{array}{l} up \rightarrow light, \neg up \rightarrow [\text{toggle}]up, \\ up \rightarrow [\text{toggle}] \neg up, \langle \text{toggle} \rangle \top \end{array} \right\} \models_{\rightsquigarrow} \text{broken} \rightarrow [\text{toggle}] \text{broken}.$$

We have $\text{toggle} \not\rightsquigarrow \neg broken$, i.e., $\neg broken$ is never caused by toggle . Hence in all contexts where $broken$ is true, after every execution of toggle , $broken$ still remains true. This independence means that the frame axiom $\text{broken} \rightarrow [\text{toggle}] \text{broken}$ is valid in the models of \rightsquigarrow .

Such a dependence-based approach has been shown [5] to subsume Reiter's solution to the frame problem [30] and moreover treats the ramification problem, even when actions with both indeterminate and indirect effects are involved [3, 16].

Definition 9 An action theory is a tuple of the form $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$.

In our running example, the corresponding action theory is

$$\mathcal{S} = \{ up \rightarrow light \}, \mathcal{E} = \{ \neg up \rightarrow [\text{toggle}]up, up \rightarrow [\text{toggle}] \neg up \}$$

$$\mathcal{X} = \{ \langle \text{toggle} \rangle \top \}, \rightsquigarrow = \left\{ \begin{array}{l} \langle \text{toggle}, light \rangle, \langle \text{toggle}, \neg light \rangle, \\ \langle \text{toggle}, up \rangle, \langle \text{toggle}, \neg up \rangle \end{array} \right\}$$

And we have $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \neg up \rightarrow [\text{toggle}] light$. (For parsimony's sake, we write $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \Phi$ instead of $\mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \models_{\rightsquigarrow} \Phi$.)

Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be an action theory and Φ a K_n -formula. $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$ denotes the action theory resulting from the contraction of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ by Φ .

Contracting a theory by a static law φ amounts to using any existing contraction operator for classical logic. Let \ominus be such an operator. Moreover, based on [19], we also need to guarantee that φ does not follow from \mathcal{E} , \mathcal{X} and \rightsquigarrow . We define contraction of a domain description by a static law as follows:

Definition 10 $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\varphi}^- = \langle \mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow \rangle$, where $\mathcal{S}^- = \mathcal{S} \ominus \varphi$ and $\mathcal{X}^- = \{ (\varphi_i \wedge \varphi) \rightarrow \langle a \rangle \top : \varphi_i \rightarrow \langle a \rangle \top \in \mathcal{X} \}$.

We now consider contraction by an executability law $\varphi \rightarrow \langle a \rangle \top$. For every executability in \mathcal{X} , we ensure that a is executable only in contexts where $\neg\varphi$ is true. The following operator does the job.

Definition 11 $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\varphi \rightarrow \langle a \rangle \top}^- = \langle \mathcal{S}, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow \rangle$, where $\mathcal{X}^- = \{ (\varphi_i \wedge \neg\varphi) \rightarrow \langle a \rangle \top : \varphi_i \rightarrow \langle a \rangle \top \in \mathcal{X} \}$.

For instance, contracting $glued \rightarrow \langle \text{toggle} \rangle \top$ in our example would give us $\mathcal{X}^- = \{ \neg glued \rightarrow \langle \text{toggle} \rangle \top \}$.

Finally, to contract a theory by $\varphi \rightarrow [a]\psi$, for every effect law in \mathcal{E} , we first ensure that a still has effect ψ whenever φ does not hold, second we enforce that a has no effect in context $\neg\varphi$ except on those literals that are consequences of $\neg\psi$. Combining this with the new dependence relation also linking a to literals involved by $\neg\psi$, we have that a may now produce $\neg\psi$ as outcome. In other words, the effect law has been contracted. The operator below formalizes this:

Definition 12 $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\varphi \rightarrow [a]\psi}^- = \langle \mathcal{S}, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow^- \rangle$, where $\rightsquigarrow^- = \rightsquigarrow \cup \{ \langle a, l \rangle : l \in \text{lit}(\text{NewCons}_{\mathcal{S}}(\neg\psi)) \}$ and $\mathcal{E}^- = \{ (\varphi_i \wedge \neg\varphi) \rightarrow [a]\psi : \varphi_i \rightarrow [a]\psi \in \mathcal{E} \} \cup \{ (\neg\varphi \wedge \neg l) \rightarrow [a]\neg l : \langle a, l \rangle \in (\rightsquigarrow^- \setminus \rightsquigarrow) \}$.

For instance, contracting the law $blackout \rightarrow [\text{toggle}] light$ from our theory would give us $\mathcal{E}^- = \{ (\neg up \wedge \neg blackout) \rightarrow [\text{toggle}]up, (up \wedge \neg blackout) \rightarrow [\text{toggle}] \neg up \}$.

5 RESULTS

We here present the main results that follow from our framework. These require the action theory under analysis to be modular [19]. An action theory is modular if a formula of a given type entailed by the whole theory can also be derived solely from its respective module (the set of formulas of the same type) with the static laws \mathcal{S} . As shown in [19], to make a domain description satisfy such a property it is enough to guarantee that there is no classical formula entailed by the theory that is not entailed by the static laws alone.

Definition 13 (Implicit static law [17]) $\varphi \in \mathfrak{Fml}$ is an implicit static law of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ if and only if $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \varphi$ and $\mathcal{S} \not\models \varphi$.

A theory is *modular* if it has no implicit static laws. Modularity of theories was originally defined in [19], but similar notions have also been addressed in the literature [4, 1, 35, 25]. A modularity-based approach for narrative reasoning about actions is given in [22].

To witness how implicit static laws can show up, consider the simple theory below, depicting the walking turkey scenario [32]:

$$\begin{aligned} \mathcal{S} &= \{\text{walking} \rightarrow \text{alive}\}, \quad \mathcal{E} = \left\{ \begin{array}{l} [\text{tease}]\text{walking}, \\ \text{loaded} \rightarrow [\text{shoot}] \neg \text{alive} \end{array} \right\} \\ \mathcal{X} &= \left\{ \begin{array}{l} \langle \text{tease} \rangle \top, \\ \langle \text{shoot} \rangle \top \end{array} \right\}, \quad \rightsquigarrow = \left\{ \begin{array}{l} \langle \text{shoot}, \neg \text{loaded} \rangle, \langle \text{shoot}, \neg \text{alive} \rangle, \\ \langle \text{shoot}, \neg \text{walking} \rangle, \langle \text{tease}, \text{walking} \rangle \end{array} \right\} \end{aligned}$$

With this description we have $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \text{alive}$.⁴ As $\mathcal{S} \not\models \text{alive}$, the formula *alive* is an implicit static law of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$.

Modular theories have several advantages [17, 15]. For example, consistency of a modular action theory can be checked by just checking consistency of \mathcal{S} : if $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ is modular, then $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \perp$ if and only if $\mathcal{S} \models \perp$. Deduction of an effect of a sequence of actions $a_1; \dots; a_n$ (prediction) needs not to take into account the effect laws for actions other than a_1, \dots, a_n . This applies in particular to plan validation when deciding whether $\langle a_1; \dots; a_n \rangle \varphi$ is the case.

Throughout this work we use multimodal logic K_n . For an assessment of the modularity principle in the Situation Calculus, see [18].

Here we show that our operators are correct w.r.t. the semantics. The first theorem establishes that the semantical contraction of models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ by Φ produces models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$.

Theorem 1 Let $\langle W, R \rangle$ be a model of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$, and let Φ be a formula that has the form of one of the three laws. For all models \mathcal{M} , if $\mathcal{M} \in \langle W, R \rangle_{\Phi}^-$, then \mathcal{M} is a model of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$.

It remains to prove that the other way round, the models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$ result from the semantical contraction of models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ by Φ . This does not hold in general, as shown by the following example: suppose there is only one atom p and one action a , and consider the theory $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ such that $\mathcal{S} = \emptyset$, $\mathcal{E} = \{p \rightarrow [a] \perp\}$, $\mathcal{X} = \{\langle a \rangle \top\}$, and $\rightsquigarrow = \emptyset$. The only model of that action theory is $\mathcal{M} = \langle \{\{\neg p\}\}, \{\{\neg p\}, \{\neg p\}\} \rangle$. By definition, $\mathcal{M}_{p \rightarrow \langle a \rangle \top}^- = \{\mathcal{M}\}$. On the other hand, $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{p \rightarrow \langle a \rangle \top}^- = \langle \emptyset, \{p \rightarrow [a] \perp\}, \{\neg p \rightarrow \langle a \rangle \top\}, \emptyset \rangle$. The contracted theory has two models: \mathcal{M} and $\mathcal{M}' = \langle \{\{p\}\}, \{\neg p\}, \{\{\neg p\}, \{\neg p\}\} \rangle$. While $\neg p$ is valid in the contraction of the models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$, it is invalid in the models of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{p \rightarrow \langle a \rangle \top}^-$.

Fortunately, we can establish a result for those action theories that are modular. The proof requires three lemmas. The first one says that for a modular theory we can restrict our attention to its ‘big’ models.

Lemma 1 Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be modular. Then $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \Phi$ if and only if $\models_{\langle W, R \rangle}^{\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle} \Phi$ for every model $\langle W, R \rangle$ of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ such that $W = \text{val}(\mathcal{S})$.

Note that the lemma does not hold for non-modular theories (the set $\{\langle W, R \rangle : \langle W, R \rangle$ is a model of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ and $W = \text{val}(\mathcal{S})\}$ is empty then).

The second lemma says that contraction preserves modularity.

⁴ Because first $\{\text{walking} \rightarrow \text{alive}, [\text{tease}]\text{walking}\} \models_{\rightsquigarrow} [\text{tease}]\text{alive}$, second $\models_{\rightsquigarrow} \neg \text{alive} \rightarrow [\text{tease}] \neg \text{alive}$ (from the independence $\text{tease} \not\rightsquigarrow \text{alive}$), and then $\mathcal{S}, \mathcal{E} \models_{\rightsquigarrow} \neg \text{alive} \rightarrow [\text{tease}] \perp$. As long as $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \langle \text{tease} \rangle \top$, we must have $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \text{alive}$.

Lemma 2 Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be modular, and let Φ be a formula of the form of one of the three laws. Then $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$ is modular.

The third one establishes the required link between the contraction operators and contraction of ‘big’ models.

Lemma 3 Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be modular, and let Φ be a formula of the form of one of the three laws. If $\mathcal{M}' = \langle \text{val}(\mathcal{S}), R' \rangle$ is a model of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$, then there is a model \mathcal{M} of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ such that $\mathcal{M}' \in \mathcal{M}_{\Phi}^-$.

Putting the three above lemmas together we get:

Theorem 2 Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be modular, Φ be a formula of the form of one of the three laws, and $\langle \mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow^- \rangle$ be $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$. If $\mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^- \models_{\rightsquigarrow^-} \Psi$, then for every model \mathcal{M} of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ and every $\mathcal{M}' \in \mathcal{M}_{\Phi}^-$ it holds that $\models^{\mathcal{M}'} \Psi$.

Our two theorems together establish correctness of the operators:

Corollary 1 Let $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ be modular, Φ be a formula of the form of one of the three laws, and $\langle \mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow^- \rangle$ be $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$. Then $\mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^- \models_{\rightsquigarrow^-} \Psi$ if and only if for every model \mathcal{M} of $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ and every $\mathcal{M}' \in \mathcal{M}_{\Phi}^-$, $\models^{\mathcal{M}'} \Psi$.

We give a sufficient condition for contraction to be successful.

Theorem 3 Let Φ be an effect or an executability law such that $\mathcal{S} \not\models_{K_n} \Phi$, and let $\langle \mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^-, \rightsquigarrow^- \rangle$ be $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^-$. If $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ is modular, then $\mathcal{S}^-, \mathcal{E}^-, \mathcal{X}^- \not\models_{\rightsquigarrow^-} \Phi$.

6 DISCUSSION AND CONCLUSION

In this work we have presented a general method for changing a domain description (alias action theory) given any formula we want to contract. We have defined a semantics for theory contraction and also presented its syntactical counterpart through contraction operators. Soundness and completeness of such operators with respect to the semantics have been established (Corollary 1).

We have also shown that modularity is a sufficient condition for a contraction to be successful (Theorem 3). This gives further evidence that the notion of modularity is fruitful.

What is the status of the AGM-postulates for contraction in our framework? First, contraction of static laws satisfies all the postulates, as soon as the underlying classical contraction operation \ominus satisfies all of them.

In the general case, however, our constructions do not satisfy the central postulate of preservation $\langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle_{\Phi}^- = \langle \mathcal{S}, \mathcal{E}, \mathcal{X}, \rightsquigarrow \rangle$ if $\mathcal{S}, \mathcal{E}, \mathcal{X} \not\models_{\rightsquigarrow} \Phi$. Indeed, suppose we have only one atom p , and a model \mathcal{M} with two worlds $w = \{p\}$ and $w' = \{\neg p\}$ such that $wR_a w'$, $w'R_a w$, and $w'R_a w'$. Then $\models^{\mathcal{M}} p \rightarrow [a] \neg p$ and $\not\models^{\mathcal{M}} [a] \neg p$, i.e., \mathcal{M} is a model of the effect law $p \rightarrow [a] \neg p$, but not of $[a] \neg p$. Now the contraction $\mathcal{M}_{[a] \neg p}^-$ yields the model \mathcal{M}' such that $R_a = W \times W$. Then $\not\models^{\mathcal{M}'} p \rightarrow [a] \neg p$, i.e., the effect law $p \rightarrow [a] \neg p$ is not preserved. Our contraction operation thus behaves rather like an update operation.

Now let us focus on the other postulates. Since our operator has a behavior which is close to the update postulate, we focus on the following basic erasure postulates introduced in [23]. Let $Cn(\mathcal{T})$ be the set of all logical consequences of a theory \mathcal{T} .

$$\mathbf{KM1} \ Cn(\langle S, E, X, \sim \rangle_{\Phi}^-) \subseteq Cn(\langle S, E, X, \sim \rangle)$$

Postulate **KM1** does not always hold because it is possible to make the formula $\varphi \rightarrow [a]\perp$ valid in the resulting theory by removing elements of R_a (cf. Definition 6).

$$\mathbf{KM2} \ \Phi \notin Cn(\langle S, E, X, \sim \rangle_{\Phi}^-)$$

Under the condition that $\langle S, E, X, \sim \rangle$ is modular, Postulate **KM2** is satisfied (cf. Theorem 3).

$$\mathbf{KM3} \text{ If } Cn(\langle S_1, E_1, X_1, \sim_1 \rangle) = Cn(\langle S_2, E_2, X_2, \sim_2 \rangle) \text{ and } \vdash_{\mathcal{R}_n} \Phi_1 \leftrightarrow \Phi_2, \text{ then } Cn(\langle S_1, E_1, X_1, \sim_1 \rangle_{\Phi_2}^-) = Cn(\langle S_2, E_2, X_2, \sim_2 \rangle_{\Phi_1}^-).$$

Theorem 4 If $\langle S_1, E_1, X_1, \sim_1 \rangle$ and $\langle S_2, E_2, X_2, \sim_2 \rangle$ are modular and the propositional contraction operator \ominus satisfies Postulate **KM3**, then Postulate **KM3** is satisfied for every $\Phi_1, \Phi_2 \in \mathfrak{Fml}$.

Following [26, 27], Eiter *et al.* [7] have investigated update of action theories in a fragment of the action description language \mathcal{C} [11] and given complexity results showing how hard such a task can be.

Update of action descriptions in their sense is always relative to some conditions (interpreted as knowledge possibly obtained from earlier observations and that should be kept). This characterizes a constraint-based update, like ours.

Even though they do not explicitly state postulates for their kind of theory update, they establish conditions for the update operator to be successful. Basically, they claim for consistency of the resulting theory; maintenance of the new knowledge and the invariable part of the description; satisfaction of the constraints; and minimal change.

With their method we can also contract by a static and an effect law. Contraction of executabilities are not explicitly addressed.

A main difference between the approach in [7] and ours is that we do not need to add new fluents at every elaboration stage: we still work on the same set of fluents, refining their behavior w.r.t. an action a . In Eiter *et al.*'s proposal an update forces changing all the variable rules appearing in the action theory by adding to each one a new update fluent. This is a constraint when elaborating action theories.

Here we have presented the case for contraction, but our definitions can be extended to revision, too. Our results can also be generalized to the case where learning new actions or fluents is involved. This means in general that more than one simple formula should be added to the belief base and must fit together with the rest of the theory with as little side-effects as possible. We are currently defining algorithms based on our operators to achieve that.

ACKNOWLEDGEMENTS

We are grateful to the anonymous referees for useful comments on an earlier version of this paper. Ivan Varzinczak has been supported by a fellowship from the government of the FEDERATIVE REPUBLIC OF BRAZIL. Grant: CAPES BEX 1389/01-7.

REFERENCES

- [1] E. Amir, '(De)composition of situation calculus theories', in *Proc. AAAI'2000*, pp. 456–463. AAAI Press/MIT Press, (2000).
- [2] M. A. Castilho, O. Gasquet, and A. Herzig, 'Formalizing action and change in modal logic I: the frame problem', *J. of Logic and Computation*, **9**(5), 701–735, (1999).
- [3] M. A. Castilho, A. Herzig, and I. J. Varzinczak, 'It depends on the context! a decidable logic of actions and plans based on a ternary dependence relation', in *NMR'02*, pp. 343–348, (2002).
- [4] L. Cholvy, 'Checking regulation consistency by using SOL-resolution', in *Proc. Int. Conf. on AI and Law*, pp. 73–79, Oslo, (1999).
- [5] R. Demolombe, A. Herzig, and I. Varzinczak, 'Regression in modal logic', *J. of Applied Non-Classical Logics*, **13**(2), 165–185, (2003).
- [6] P. Doherty, W. Łukaszewicz, and E. Madalinska-Bugaj, 'The PMA and relativizing change for action update', in *Proc. KR'98*, pp. 258–269. Morgan, (1998).
- [7] T. Eiter, E. Erdem, M. Fink, and J. Senko, 'Updating action domain descriptions', in *Proc. IJCAI'05*, pp. 418–423. Morgan, (2005).
- [8] N. Y. Foo and D. Zhang, 'Dealing with the ramification problem in the extended propositional dynamic logic', in *Advances in Modal Logic*, volume 3, 173–191, World Scientific, (2002).
- [9] K. D. Forbus, 'Introducing actions into qualitative simulation', in *Proc. IJCAI'89*, pp. 1273–1278. Morgan, (1989).
- [10] P. Gärdenfors, *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, MIT Press, 1988.
- [11] M. Gelfond and V. Lifschitz, 'Action languages', *ETAI*, **2**(3–4), 193–210, (1998).
- [12] S. O. Hansson, *A Textbook of Belief Dynamics: Theory Change and Database Updating*, Kluwer, 1999.
- [13] D. Harel, 'Dynamic logic', in *Handbook of Philosophical Logic*, volume II, 497–604, D. Reidel, Dordrecht, (1984).
- [14] A. Herzig and O. Rifi, 'Propositional belief base update and minimal change', *Artificial Intelligence*, **115**(1), 107–138, (1999).
- [15] A. Herzig and I. Varzinczak. Metatheory of actions: beyond consistency. To appear.
- [16] A. Herzig and I. Varzinczak, 'An assessment of actions with indeterminate and indirect effects in some causal approaches', Technical Report 2004–08-R, IRIT – Université Paul Sabatier, (May 2004).
- [17] A. Herzig and I. Varzinczak, 'Domain descriptions should be modular', in *Proc. ECAI'04*, pp. 348–352. IOS Press, (2004).
- [18] A. Herzig and I. Varzinczak, 'Cohesion, coupling and the meta-theory of actions', in *Proc. IJCAI'05*, pp. 442–447. Morgan, (2005).
- [19] A. Herzig and I. Varzinczak, 'On the modularity of theories', in *Advances in Modal Logic*, volume 5, 93–109, King's College, (2005).
- [20] K. Inoue, 'Linear resolution for consequence finding', *Artificial Intelligence*, **56**(2–3), 301–353, (1992).
- [21] Y. Jin and M. Thielscher, 'Iterated belief revision, revised', in *Proc. IJCAI'05*, pp. 478–483. Morgan, (2005).
- [22] A. Kakas, L. Michael, and R. Miller, 'Modular- \mathcal{E} : an elaboration tolerant approach to the ramification and qualification problems', in *Proc. LPNR'05*, pp. 211–226. Springer, (2005).
- [23] H. Katsuno and A. O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (1991).
- [24] H. Katsuno and A. O. Mendelzon, 'On the difference between updating a knowledge base and revising it', in *Belief revision*, ed.. P. Gärdenfors, 183–203, Cambridge, (1992).
- [25] J. Lang, F. Lin, and P. Marquis, 'Causal theories of action – a computational core', in *Proc. IJCAI'03*, pp. 1073–1078. Morgan, (2003).
- [26] R. Li and L.M. Pereira, 'What is believed is what is explained', in *Proc. AAAI'96*, pp. 550–555. AAAI Press/MIT Press, (1996).
- [27] P. Liberatore, 'A framework for belief update', in *Proc. JELIA'2000*, pp. 361–375, (2000).
- [28] J. McCarthy, *Mathematical logic in artificial intelligence*, Daedalus, 1988.
- [29] S. Popkorn, *First Steps in Modal Logic*, Cambridge, 1994.
- [30] R. Reiter, 'The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression', in *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, 359–380, Academic Press, (1991).
- [31] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. J. Levesque, 'Iterated belief change in the situation calculus', in *Proc. KR'2000*, pp. 527–538. Morgan, (2000).
- [32] M. Thielscher, 'Computing ramifications by postprocessing', in *Proc. IJCAI'95*, pp. 1994–2000. Morgan, (1995).
- [33] M.-A. Winslett, 'Reasoning about action using a possible models approach', in *Proc. AAAI'88*, pp. 89–93. Morgan, (1988).
- [34] M.-A. Winslett, 'Updating logical databases', in *Handbook of Logic in AI and Logic Programming*, volume 4, 133–174, Oxford, (1995).
- [35] D. Zhang, S. Chopra, and N. Y. Foo, 'Consistency of action descriptions', in *Proc. PRICAI'02*. Springer, (2002).
- [36] D. Zhang and N. Y. Foo, 'EPDL: A logic for causal reasoning', in *Proc. IJCAI'01*, pp. 131–138. Morgan, (2001).

On the Logic of Theory Change: Relations between Incision and Selection Functions

Marcelo A. Falappa¹ and Eduardo L. Fermé² and Gabriele Kern-Isbner³

Abstract. This work elaborates on the connection between partial meet contractions and kernel contractions in belief change theory. We present a way to define incision functions (used in kernel contractions) from selection functions (used in partial meet contractions) and vice versa. Then we make precise under which conditions there are exact correspondences between selection and incision functions so that the same contraction operations can be obtained by using either of them.

Key Words: Belief Revision, AGM Theory, Partial Meet Contractions, Kernel Contractions.

1 Introduction

Belief revision systems are logical frameworks for modelling the dynamics of knowledge, that is, how agents modify their beliefs when they receive new information. The main problem arises when that information is inconsistent with the beliefs that the agent holds in its epistemic state. In order to accept the new information, the agent will have to give up some information while preserving as much of the old information as possible.

The AGM-framework [1] is the most popular framework to guide the change of belief. It distinguishes between three kinds of changes, namely expansion, contraction and revision. Since expansion deals with a simple case of revision, and revision can be based on contraction, the contraction operators are considered to be particularly interesting. Basically, AGM contractions can be realized by *partial meet contractions*, which are based on a selection among subsets of the original set that do not imply the information to be retracted via so-called *selection functions*. On the other hand, *kernel contractions* [8] are based on a selection among the sentences that are relevant to derive the sentence to be retracted. In order to perform a contraction, kernel contractions use *incision functions* which cut into the minimal subsets that imply the information to be given up.

There are other change operators, which also can be defined in similar ways by making use of selection or incision functions, respectively: For instance, a *consolidation* [9] that makes a contraction of a set of sentences by a contradiction; a *semi-revision* [9] that allows a non-prioritized revision of a set of sentences by a single sentence, and a *revision by sets of sentences* [6, 7] that makes a non-prioritized revision of a set of sentences by a set of sentences. In all these cases, every partial meet operator is a kernel operator, but not vice versa.

In this paper, we elaborate the relation between partial meet and kernel contraction in more depth. Since every partial meet operator is defined in terms of a selection function and every kernel operator is defined in terms of an incision function, we investigate in particular, how these functions can be related to one another more clearly. We show how incision functions can be obtained from selection functions and vice versa. In this way, bidirectional mappings between these two kinds of functions are given by which partial meet and kernel contractions can be directly related. As the main results of this paper, we make precise under which conditions applying one mapping after the other yields the identical contraction operator (more exactly, the identical selection or incision function, respectively).

The main focus of this paper is on changing *belief bases*, for several reasons. First, since handling belief sets (*i.e.*, deductively closed sets of sentences) is computationally not feasible, taking finite or countable belief bases as convenient representations of beliefs is much more natural for practical implementations of belief revision. Second, making use of belief bases allows for distinguishing between explicitly represented beliefs and implicitly derived beliefs; so, we do not insist on the *principle of irrelevance of syntax*. Considering belief bases can even help distinguishing between different inconsistent belief sets. And finally, for belief bases, kernel contractions and partial meet contractions are more loosely connected than for belief sets, so studying the links between both types of revision operations is more rewarding. On the other hand, kernel contractions on belief bases can be, in a sense, related to truth maintenance systems and to argumentation systems [3, 4, 5, 11].

The organization of this paper is as follows: In section 2, we briefly recall the basic definitions of partial meet and kernel contraction the relations between which we start to investigate in section 3. Sections 4 and 5 show how to define incision functions from selection functions and vice versa. After that, the grounds are laid to tighten these bands by revealing exact correspondences between partial meet and kernel contractions in section 6. We summarize our results in 7.

2 Different kinds of contraction

In the following, let K be a set of sentences of a propositional language \mathcal{L} which represents the knowledge base of an intelligent agent. We will recall some facts about the two kinds of contraction which are in the focus of this paper.

2.1 Partial meet contractions

Partial meet contractions are exactly the basic AGM-contractions. They are based on the concept of a remainder set.

¹ Artificial Intelligence Research and Development Laboratory, Department of Computer Science and Engineering, Universidad Nacional del Sur, CONICET, Bahía Blanca, Argentina, email: mfalappa@cs.uns.edu.ar

² Departamento de Matemática e Engenharias, Universidade da Madeira, Portugal, email: ferme@uma.pt

³ Department of Computer Science, University of Dortmund, Dortmund, Germany, email: gabriele.kern-isbner@cs.uni-dortmund.de

Definition 1 ([2]) Let K be a set of sentences and α a sentence. Then $K^\perp\alpha$ is the set of all H such that $H \subseteq K$, $H \not\models \alpha$ and if $H \subset K' \subseteq K$, $H \neq K'$, then $K' \vdash \alpha$. The set $K^\perp\alpha$ is called the remainder set of K with respect to α , and its elements are called the α -remainders of K .

In order to define a partial meet contraction operator, we need a selection function. This function makes a selection among the α -remainders, choosing those candidate sets for contraction which are preferred in some sense.

Definition 2 Let K be a set of sentences. A selection function for K is a function “ γ ” ($\gamma : 2^{\mathcal{L}} \Rightarrow 2^{\mathcal{L}}$) such that for any sentence $\alpha \in \mathcal{L}$, it holds that:

- 1) If $K^\perp\alpha \neq \emptyset$, then $\gamma(K^\perp\alpha)$ is a non-empty subset of $K^\perp\alpha$.
- 2) If $K^\perp\alpha = \emptyset$, then $\gamma(K^\perp\alpha) = K$.

Definition 3 Given a set of sentences K , a sentence α and a selection function γ for K , the partial meet contraction of K by α , denoted by $K \dot{\div}_\gamma \alpha$, is defined as:

$$K \dot{\div}_\gamma \alpha = \cap \gamma(K^\perp\alpha)$$

If γ selects a single element, then the induced contraction is called a maxichoice contraction.

That is, $K \dot{\div}_\gamma \alpha$ is equal to the intersection of the α -remainders of K selected by γ .

Obviously, for each $H \in \gamma(K^\perp\alpha)$, we have $\cap \gamma(K^\perp\alpha) \subseteq H$. Selection functions which cover all of these remainder sets are called complete.

Definition 4 ([1]) A partial meet selection function γ on K is complete iff whenever $\cap \gamma(K^\perp\alpha) \subseteq H$ for $H \in K^\perp\alpha$, then $H \in \gamma(K^\perp\alpha)$.

Important examples for belief bases that give rise to complete selection functions, are belief sets (i.e., $K = Cn(K)$).

Proposition 1 ([1]) If the language \mathcal{L} is finite and K is a belief set, then any selection function γ on K is complete.

2.2 Kernel contractions

Kernel sets are those subsets of the knowledge base which are relevant for the information to be contracted.

Definition 5 ([8]) Let K be a set of sentences and α a sentence. Then $K^\perp\alpha$ is the set of all X such that $X \subseteq K$, $X \vdash \alpha$, and if $K' \subset X$, then $K' \not\models \alpha$. The set $K^\perp\alpha$ is called the kernel set, and its elements are called the α -kernels of K .

In order to retract some information α from K , we have to give up sentences in each α -kernel. This is done by incision functions.

Definition 6 Let K be a set of sentences. An incision function for K is a function “ σ ” ($\sigma : 2^{\mathcal{L}} \Rightarrow 2^{\mathcal{L}}$) such that for any sentence $\alpha \in \mathcal{L}$, the following hold:

- 1) $\sigma(K^\perp\alpha) \subseteq \cup(K^\perp\alpha)$.
 - 2) If $X \in K^\perp\alpha$ and $X \neq \emptyset$ then $(X \cap \sigma(K^\perp\alpha)) \neq \emptyset$.
- In the limit case when $K^\perp\alpha = \emptyset$ then $\sigma(K^\perp\alpha) = \emptyset$.

So, incision functions cut into each α -kernel, removing at least one sentence. Since all α -kernels are minimal subsets implying α , from the resulting sets it is no longer possible to derive α . Hence, incision functions may be used to derive contraction operations.

Definition 7 Given a set of sentences K , a sentence α and an incision function σ for K , the kernel contraction of K by α , denoted by $K \dot{\div}_\sigma \alpha$, is defined as:

$$K \dot{\div}_\sigma \alpha = K \setminus \sigma(K^\perp\alpha)$$

That is, $K \dot{\div}_\sigma \alpha$ can be obtained by erasing from K the sentences cut out by σ .

As Hansson observed [10], kernel operators are a very general class of contraction operators, and it is easy to find examples in which the behavior of kernel operations are not at all intuitively plausible. For instance, let p and q be two logically independent sentences, and let:

$$K = \{p, p \vee q, p \leftrightarrow q\}.$$

Then we have that:

$$K^\perp(p \wedge q) = \{\{p, p \leftrightarrow q\}, \{p \vee q, p \leftrightarrow q\}\}.$$

Suppose that $\sigma(K^\perp(p \wedge q)) = \{p \vee q, p \leftrightarrow q\}$. This gives rise to an operator of contraction such that:

$$K \dot{\div}_\sigma(p \wedge q) = K \setminus \sigma(K^\perp(p \wedge q)) = \{p\}.$$

It does not seem sensible for $p \vee q$ to be given up here. Since p was not given up, $p \vee q$ is implied by the contracted set $K \dot{\div}_\sigma(p \wedge q)$. The exclusion of $p \vee q$ was unnecessary, and it violates the basic principle of minimality: nothing should be given up without reason [10].

Therefore, further properties are necessary to improve the well-behavedness of kernel contractions.

Definition 8 ([8, 10]) An incision function σ for a set K is smooth if and only if it holds for all subsets K' of K that: if $K' \vdash \beta$ and $\beta \in \sigma(K^\perp\alpha)$ then $K' \cap \sigma(K^\perp\alpha) \neq \emptyset$.

This property is needed to ensure that a kernel contraction operator $\dot{\div}_\sigma$ satisfies the postulate of relative closure [10]: $K \cap Cn(K \dot{\div}_\sigma \alpha) \subseteq K \dot{\div}_\sigma \alpha$.

3 Relating partial meet and kernel contractions

All partial meet contractions are kernel contractions, but in general, these two types of contractions are different – not every kernel contraction can be realized as a partial meet contraction. Not even smoothness (cf. Definition 8) can guarantee kernel contractions to be the same as partial meet contractions [10]. However, it is well-known that for belief sets, smooth kernel contractions and partial meet contractions coincide (cf. [8]).

A direct link from incision functions to remainder sets can be established by considering minimal incision functions.

Definition 9 An incision function σ for a set K is minimal if no proper subset of $\sigma(K^\perp\alpha)$ defines an incision function.

As the following proposition shows, the cut-ins of minimal incision functions are minimal with respect to at least one α -kernel.

Proposition 2 Let σ be a minimal incision function, and $x \in \sigma(K^\perp\alpha)$. Then there is an $X \in K^\perp\alpha$ such that $X \cap \sigma(K^\perp\alpha) = \{x\}$.

Proof. Assume the contrary: For all $X \in K^{\perp\perp}\alpha$ such that $x \in X$, it holds that $|X \cap \sigma(K^{\perp\perp}\alpha)| > 1$. Then define $\sigma'(K^{\perp\perp}\alpha) = \sigma(K^{\perp\perp}\alpha) \setminus \{x\}$. For $X \in K^{\perp\perp}\alpha$ with $x \notin X$, we have $X \cap \sigma'(K^{\perp\perp}\alpha) = X \cap \sigma(K^{\perp\perp}\alpha) \neq \emptyset$. For $X \in K^{\perp\perp}\alpha$ with $x \in X$, we have $|X \cap \sigma'(K^{\perp\perp}\alpha)| \geq 1$, hence $X \cap \sigma'(K^{\perp\perp}\alpha) \neq \emptyset$. Therefore, σ' is an incision function, which contradicts the presupposition that σ is minimal. \square

It turns out that minimality of incision corresponds to maximality of contraction. First, the following proposition shows that each kernel contraction defined by a minimal incision function satisfies *fullness*:

Fullness: If $\beta \in K$ and $\beta \notin K \div \alpha$ then $((K \div \alpha) \cup \{\beta\}) \vdash \alpha$.

Proposition 3 Let \div_σ be a kernel contraction for K based on an incision function σ . If σ satisfies minimality then \div_σ satisfies fullness.

Proof. Let σ be “minimal”. Let $\beta \in K$ and $\beta \notin K \div \alpha$. It follows by definition of kernel contraction that $\beta \in \sigma(K^{\perp\perp}\alpha)$. Assume by *reductio* that $((K \div \alpha) \cup \{\beta\}) \not\vdash \alpha$. Then $K \setminus (\sigma(K^{\perp\perp}\alpha) \setminus \{\beta\}) \not\vdash \alpha$. Let $\sigma'(K^{\perp\perp}\alpha) = \sigma(K^{\perp\perp}\alpha) \setminus \{\beta\}$. It is easy to prove that σ' is an incision function. Hence, σ is not minimal (absurd). \square

Any operator of kernel contraction satisfies the properties *success*, *inclusion* and *uniformity* (cf. e.g. [10]). Together with *fullness*, these properties characterize *maxichoice contractions* (cf. [10]). So we have the following corollary:

Corollary 1 Let \div_σ be a kernel contraction for K based on an incision function σ . If σ satisfies minimality then \div_σ is a maxichoice contraction for K .

Proposition 4 For all $H \subseteq K$, $H \in K^\perp\alpha$ iff there is a minimal incision function σ such that $H = K \setminus \sigma(K^{\perp\perp}\alpha)$.

Proof. Let $H \in K^\perp\alpha$. Define σ via $\sigma(K^{\perp\perp}\alpha) = K \setminus H$; to be shown: σ is a minimal incision function.

Let $x \in \sigma(K^{\perp\perp}\alpha) = K \setminus H$. Then $H \cup \{x\} \vdash \alpha$, so there is $X \in K^{\perp\perp}\alpha$ such that $X \subseteq H \cup \{x\}$. Then it must be that $x \in X$ (otherwise, we would have $X \subseteq H$), hence $\sigma(K^{\perp\perp}\alpha) \subseteq \bigcup K^{\perp\perp}\alpha$. Now choose $X \in K^{\perp\perp}\alpha$ arbitrarily. Then there must be $x \in X$ with $x \notin H$ (otherwise $X \subseteq H$, a contradiction). Thus $X \cap K \setminus H \neq \emptyset$. So, σ is an incision function. To see that it is minimal, assume there is an incision function σ_1 with $\sigma_1(K^{\perp\perp}\alpha) \subsetneq \sigma(K^{\perp\perp}\alpha) = K \setminus H$. Then $H \subsetneq K \setminus \sigma_1(K^{\perp\perp}\alpha)$, and $K \setminus \sigma_1(K^{\perp\perp}\alpha) \not\vdash \alpha$. This contradicts the maximality of H .

Conversely, let σ be a minimal incision function, and set $H = K \setminus \sigma(K^{\perp\perp}\alpha)$. Then certainly, $H \not\vdash \alpha$. Choose $x \in K$, $x \notin H$ arbitrarily. Then $x \in \sigma(K^{\perp\perp}\alpha)$, and by Proposition 2, there is $X \in K^{\perp\perp}\alpha$ such that $X \cap \sigma(K^{\perp\perp}\alpha) = \{x\}$. Then $X \setminus \{x\} \subseteq K \setminus \sigma(K^{\perp\perp}\alpha) = H$, hence $X \subseteq H \cup \{x\}$, and therefore $H \cup \{x\} \vdash \alpha$. This proves that $H \in K^\perp\alpha$. \square

So, minimal incision functions “generate” the remainder sets of K .

In the rest of this paper, we will study the relationships between these two kinds of contraction operators by investigating how incision functions can be obtained from selection functions and vice versa. The following proposition will prove useful for that purpose.

Proposition 5 ([10]) The following conditions are equivalent:

1. $K^\perp\alpha = K^\perp\beta$.
2. $K^{\perp\perp}\alpha = K^{\perp\perp}\beta$.
3. For all subset H of K : $H \vdash \alpha$ if and only if $H \vdash \beta$.

4 Incision functions from selection functions

We will present a formal definition of an incision function from a selection function.

Definition 10 Let γ be a selection function for a set K . We define an associated incision function σ_γ as:

$$\sigma_\gamma(K^{\perp\perp}\alpha) =_{\text{def}} K \setminus \gamma(K^\perp\alpha)$$

The intuition behind this definition is the following: β is selected by an incision function σ (over the set of α -kernels of K) if and only if β does not belong to some α -remainder of K selected by γ .

Example. Let $K = \{p, r, s, p \rightarrow q, r \rightarrow q, \neg t \rightarrow \neg q\}$. Then $K^\perp q = \{\{p, r, s, \neg t \rightarrow \neg q\}, \{p, s, r \rightarrow q, \neg t \rightarrow \neg q\}, \{s, p \rightarrow q, r \rightarrow q, \neg t \rightarrow \neg q\}, \{r, s, p \rightarrow q, \neg t \rightarrow \neg q\}, \{r, s, p \rightarrow q, \neg t \rightarrow \neg q\}\}$. Suppose that $\gamma(K^\perp q) = \{\{p, r, s, \neg t \rightarrow \neg q\}, \{r, s, p \rightarrow q, \neg t \rightarrow \neg q\}\}$. Then the associated incision function is $\sigma_\gamma(K^{\perp\perp} q) = \sigma_\gamma(\{p, p \rightarrow q\}, \{r, r \rightarrow q\}) = \{p, p \rightarrow q, r \rightarrow q\}$.

We have to prove that indeed, the function σ_γ satisfies the properties of an incision function.

Proposition 6 The function σ_γ presented in Definition 10 is an incision function.

Proof.

1. σ_γ is a well defined function due to Proposition 5.
2. Next, we have to show that $\sigma_\gamma(K^{\perp\perp}\alpha) \subseteq \bigcup(K^{\perp\perp}\alpha)$.
If $\vdash \alpha$, then $K^\perp\alpha = \emptyset$ and it follows by definition of γ that $\gamma(K^\perp\alpha) = \{K\}$. If $K \not\vdash \alpha$ it follows by definition of γ that $\gamma(K^\perp\alpha) = \{K\}$. In both cases $\sigma_\gamma(K^{\perp\perp}\alpha) = \emptyset$ and we are done.
Let $\not\vdash \alpha$ and $K \vdash \alpha$. Then $K \setminus \gamma(K^\perp\alpha) \neq \emptyset$. Let $\delta \in K$ and $\delta \notin \bigcup \gamma(K^\perp\alpha)$ (i.e., $\delta \in \sigma_\gamma(K^{\perp\perp}\alpha)$). Then there exists a $H \in \gamma(K^\perp\alpha)$ such that $\delta \notin H$ and $H \not\vdash \alpha$. Let $H' = H \cup \{\delta\}$. It follows by definition of $K^\perp\alpha$ that $H' \vdash \alpha$. By compactness there exists a minimal subset H'' of H' such that $H'' \vdash \alpha$ and $\delta \in H''$. Hence $H'' \in K^{\perp\perp}\alpha$ and $\delta \in \bigcup(K^{\perp\perp}\alpha)$.
3. Finally, it must be shown that if $X \in K^{\perp\perp}\alpha$ and $X \neq \emptyset$ then $(X \cap \sigma_\gamma(K^{\perp\perp}\alpha)) \neq \emptyset$.
Let $X \in K^{\perp\perp}\alpha$ and $X \neq \emptyset$. Let $\delta \in X$ and assume by *reductio* that $(X \cap \sigma_\gamma(K^{\perp\perp}\alpha)) = \emptyset$. Then by definition of σ_γ , $(X \cap (K \setminus \gamma(K^\perp\alpha))) = \emptyset$. Due to $X \subseteq K$ it follows that $X \subseteq \bigcup \gamma(K^\perp\alpha)$. Let $H \in \gamma(K^\perp\alpha)$. Then $X \subseteq H$ and hence $H \vdash \alpha$ (absurd). \square

5 Selection functions from incision functions

Also, we may define selection functions from incision functions.

Definition 11 Let σ a incision function for a set K . We may define an associated selection function γ_σ on $K^\perp\alpha$ by:

$$\gamma_\sigma(K^\perp\alpha) = \begin{cases} \{H \in K^\perp\alpha : (K \setminus \sigma(K^{\perp\perp}\alpha)) \subseteq H\}, & \text{if } K^\perp\alpha \neq \emptyset \\ \{K\}, & \text{otherwise} \end{cases}$$

Again, we have to show that this definition makes sense, i.e., that indeed, γ_σ fulfills the required properties.

Proposition 7 The function γ_σ presented in Definition 11 is a selection function.

Proof.

1. γ_σ is a well defined function: Suppose that $K^\perp\alpha = K^\perp\beta$. We must show that $\gamma_\sigma(K^\perp\alpha) = \gamma_\sigma(K^\perp\beta)$. By Proposition 5 $K^\perp\alpha = K^\perp\beta$ and the rest follows trivially.
2. By definition, it holds that if $K^\perp\alpha = \emptyset$ then $\gamma_\sigma(K^\perp\alpha) = K$.
3. Again by definition, $\gamma_\sigma(K^\perp\alpha) \subseteq K^\perp\alpha$. Let $K^\perp\alpha \neq \emptyset$; we have to show that $\gamma_\sigma(K^\perp\alpha) \neq \emptyset$, too.. From $K^\perp\alpha \neq \emptyset$, it follows that $\nexists \alpha$. We have two cases:
 - (a) $K \not\nvdash \alpha$. Then $K \in K^\perp\alpha$, and due to $(K \setminus \sigma(K^\perp\alpha)) \subseteq K$ it follows that $K \in \gamma_\sigma(K^\perp\alpha)$.
 - (b) $K \vdash \alpha$. By $(K \setminus \sigma(K^\perp\alpha)) \not\nvdash \alpha$ then there exists $H' \in K^\perp\alpha$ such that $(K \setminus \sigma(K^\perp\alpha)) \subseteq H'$. Hence $H' \in \gamma_\sigma(K^\perp\alpha)$. \square

Please note that this same construction of a selection function from an incision function, as we put it in Definition 11, is used by Hansson in [8] to prove that for belief sets, smooth kernel contractions are the same as partial meet contractions.

6 Bidirectional mappings

Having defined selection functions from incision functions and incision functions from selection functions, we will now investigate when these circles close, *i.e.*, under which conditions we obtain the same function that we started with after going forth and back. We will first consider selection functions.

Proposition 8 *Let γ be a selection function of a set K , σ_γ the incision function defined from γ using the Definition 10, and γ' the selection function defined from σ_γ using the Definition 11, *i.e.*, $\gamma' = \gamma_{\sigma_\gamma}$. Then:*

1. $\gamma(K^\perp\alpha) \subseteq \gamma'(K^\perp\alpha)$.
2. $\cap\gamma'(K^\perp\alpha) = \cap\gamma(K^\perp\alpha)$.

Proof. By definition, we have

$$\begin{aligned} \gamma'(K^\perp\alpha) &= \{H \in K^\perp\alpha : K \setminus \sigma_\gamma(K^\perp\alpha) \subseteq H\} \\ &= \{H \in K^\perp\alpha : K \setminus (K \setminus \cap\gamma(K^\perp\alpha)) \subseteq H\} \\ &= \{H \in K^\perp\alpha : \cap\gamma(K^\perp\alpha) \subseteq H\} \end{aligned} \quad (1)$$

1. Let $H' \in \gamma(K^\perp\alpha)$. Then $\cap\gamma(K^\perp\alpha) \subseteq H'$. Hence $\gamma(K^\perp\alpha) \subseteq \gamma'(K^\perp\alpha)$.
2. It follows immediately from 1., that $\cap\gamma'(K^\perp\alpha) \subseteq \cap\gamma(K^\perp\alpha)$. The other inclusion is straightforward, too: For each $H \in \gamma'(K^\perp\alpha)$, we have $\cap\gamma(K^\perp\alpha) \subseteq H$, due to equation (1), so $\cap\gamma(K^\perp\alpha) \subseteq \cap\gamma'(K^\perp\alpha)$. \square

In the following, we will focus on conditions when $\gamma'(K^\perp\alpha) = \gamma(K^\perp\alpha)$ holds.

Proposition 9 *Let $\gamma, \sigma_\gamma, \gamma'$ be as in Proposition 8. Then γ' is complete.*

Proof. Let $H \in K^\perp\alpha$ such that $\cap\gamma'(K^\perp\alpha) \subseteq H$. By Proposition 8, 2., then also $\cap\gamma(K^\perp\alpha) \subseteq H$, which yields $H \in \gamma'(K^\perp\alpha)$. This proves the completeness of γ' . \square

So $\gamma' = \gamma_{\sigma_\gamma}$ can be regarded as the *completion* of the selection function γ . The following Proposition summarizes that completeness is a necessary and sufficient condition for the circle starting from selection functions to close.

Proposition 10 *Let $\gamma, \sigma_\gamma, \gamma'$ be as in Proposition 8. Then $\gamma'(K^\perp\alpha) = \gamma(K^\perp\alpha)$ iff γ is complete.*

Proof. $\gamma(K^\perp\alpha) \subseteq \gamma'(K^\perp\alpha)$ holds in general, due to Proposition 8, 1. Let $H \in \gamma'(K^\perp\alpha)$; by (1), this means that $H \in K^\perp\alpha$ and $\cap\gamma(K^\perp\alpha) \subseteq H$. For each such H to be in $\gamma(K^\perp\alpha) \subseteq H$ is equivalent to the completeness of γ . \square

Propositions 1 and 10 immediately yield the following:

Corollary 2 *If the language \mathcal{L} is finite and K is a belief set, then $\gamma'(K^\perp\alpha) = \gamma(K^\perp\alpha)$.*

We will now focus on constructions that begin with incision functions. As in the previous case, one of the inclusions is straightforward.

Proposition 11 *Let σ be an incision function, γ_σ the selection function defined from σ using the Definition 11, and σ' the incision function defined from γ_σ using the Definition 10, *i.e.*, $\sigma' = \sigma_{\gamma_\sigma}$. Then it holds that $\sigma'(K^\perp\alpha) \subseteq \sigma(K^\perp\alpha)$.*

Proof. By definition, we have:

$$\sigma'(K^\perp\alpha) = K \setminus \cap\{H \in K^\perp\alpha : K \setminus \sigma(K^\perp\alpha) \subseteq H\}.$$

Suppose that $\beta \in \sigma'(K^\perp\alpha)$. Then there is $H' \in K^\perp\alpha$ such that $K \setminus \sigma(K^\perp\alpha) \subseteq H'$ and $\beta \notin H'$. Then $\beta \notin K \setminus \sigma(K^\perp\alpha)$. Since $\beta \in K$ then $\beta \in \sigma(K^\perp\alpha)$. \square

The following property is a straightforward rephrasing of the well-known relevance postulate, formulated by Hansson [10] for incision functions:

Relevance: For all $\beta \in \sigma(K^\perp\alpha)$, there is an X such that $(K \setminus \sigma(K^\perp\alpha)) \subseteq X \subseteq K$, $X \nvDash \alpha$ and $X \cup \{\beta\} \vdash \alpha$.

Proposition 12 *Let σ be an incision function and σ' be as defined in Proposition 11. Then it holds that:*

$$\sigma(K^\perp\alpha) \subseteq \sigma'(K^\perp\alpha)$$

if and only if σ satisfies relevance.

Proof. We will prove the statement in its contrapositive form, *i.e.*, $\sigma(K^\perp\alpha) \not\subseteq \sigma'(K^\perp\alpha)$ iff there is $\beta \in \sigma(K^\perp\alpha)$ such that for all X , $(K \setminus \sigma(K^\perp\alpha)) \subseteq X \subseteq K$ and $X \nvDash \alpha$ implies that $X \cup \{\beta\} \nvDash \alpha$.

First, let $\sigma(K^\perp\alpha) \not\subseteq \sigma'(K^\perp\alpha)$. Then there is $\beta \in \sigma(K^\perp\alpha)$ such that $\beta \notin \sigma'(K^\perp\alpha)$, *i.e.*, there is $\beta \in \sigma(K^\perp\alpha)$ such that for all $H \in K^\perp\alpha$ with $(K \setminus \sigma(K^\perp\alpha)) \subseteq H$, it holds that $\beta \in H$. Now, let $X \subseteq K$, $(K \setminus \sigma(K^\perp\alpha)) \subseteq X$ and $X \nvDash \alpha$. Then there is a maximal $H \in K^\perp\alpha$ with $X \subseteq H$, so $(K \setminus \sigma(K^\perp\alpha)) \subseteq X \subseteq H$ and hence $\beta \in H$. But then also $X \cup \{\beta\} \subseteq H$, and therefore $X \cup \{\beta\} \nvDash \alpha$, since $H \in K^\perp\alpha$.

To prove sufficiency, assume there is $\beta \in \sigma(K^\perp\alpha)$ such that for all X , $(K \setminus \sigma(K^\perp\alpha)) \subseteq X \subseteq K$ and $X \nvDash \alpha$ implies $X \cup \{\beta\} \nvDash \alpha$. Let $H \in K^\perp\alpha$ with $(K \setminus \sigma(K^\perp\alpha)) \subseteq H$. Then $H \nvDash \alpha$ and hence, by assumption, $H \cup \{\beta\} \nvDash \alpha$. But since H is an α -remainder and hence maximal, this means that $\beta \in H$. Therefore, $\beta \in \cap\{H \in K^\perp\alpha : (K \setminus \sigma(K^\perp\alpha)) \subseteq H\}$ and consequently, $\beta \notin \sigma'(K^\perp\alpha)$. \square

So, *relevance* is a necessary and sufficient condition for an incision function σ to guarantee that $\sigma = \sigma_{\gamma_\sigma}$. The following two propositions clarify the relationships between *relevance* and *smoothness*.

Proposition 13 If K is a belief set and σ is smooth, then it satisfies relevance.

Proof. Let $\beta \in \sigma(K^\perp\alpha)$. Assume by reductio that $(\neg\alpha \vee \beta) \in \sigma(K^\perp\alpha)$. It follows that there exists $H \in K^\perp\alpha$ such that $H \vdash \alpha$ and $H \setminus \{\neg\alpha \vee \beta\} \not\vdash \alpha$. Clearly, $(H \setminus \{\neg\alpha \vee \beta\}) \cup \{\neg\alpha \vee \beta\} \vdash \alpha$. By deduction theorem $(H \setminus \{\neg\alpha \vee \beta\}) \vdash (\neg\alpha \vee \beta) \rightarrow \alpha$. Hence $(H \setminus \{\neg\alpha \vee \beta\}) \vdash \alpha$. From this absurd we can conclude that $(\neg\alpha \vee \beta) \notin \sigma(K^\perp\alpha)$.

Let $H' = \{\neg\alpha \vee \beta, \alpha \vee \beta\}$. Since $H' \vdash \beta$ and $\beta \in \sigma(K^\perp\alpha)$, smoothness yields $H' \cap \sigma(K^\perp\alpha) \neq \emptyset$. Hence $(\alpha \vee \beta) \in \sigma(K^\perp\alpha)$. Smoothness yields also that $K \setminus \sigma(K^\perp\alpha) \not\vdash (\alpha \vee \beta)$. Let $H'' = (K \setminus \sigma(K^\perp\alpha)) \cup \{\alpha \vee \neg\beta\}$. H'' satisfies the conditions required for relevance. \square

Proposition 14 If σ satisfies relevance then it is smooth.

Proof. Assume that σ is not smooth. Then there exists H such that $H \subseteq K$, $H \vdash \beta$, $\beta \in \sigma(K^\perp\alpha)$ and such that $H \cap \sigma(K^\perp\alpha) = \emptyset$. Then $K \setminus \sigma(K^\perp\alpha) \vdash \beta$, from which it follows that relevance does not hold. \square

The following corollary is now immediate:

Corollary 3 If K is a belief set, then an incision function is smooth iff it satisfies relevance.

So, smoothness and relevance are closely related. However, smoothness is not enough to yield $\sigma(K^\perp\alpha) \subseteq \sigma'(K^\perp\alpha)$ and hence to close the circle (see Proposition 11), as the following example illustrates.

Example. Let p, q and r be logically independent sentences, and let $K = \{p, q, r\}$. Suppose that we want to contract K by $p \wedge (q \vee r)$. Then we have:

$$K^\perp(p \wedge (q \vee r)) = \{\{p, q\}, \{p, r\}\}.$$

Let σ be such that $\sigma(K^\perp(p \wedge (q \vee r))) = \{p, r\}$. Then $K \div_\sigma (p \wedge (q \vee r)) = \{q\}$. This kernel contraction is smooth. Hence, we have that $\sigma'(K^\perp(p \wedge (q \vee r))) = K \setminus \{q, r\} = \{p\}$. This means that:

$$\sigma(K^\perp(p \wedge (q \vee r))) = \{p, r\} \not\subseteq \{p\} = \sigma'(K^\perp(p \wedge (q \vee r))).$$

7 Conclusion

In this paper, we have investigated in detail the relations between partial meet and kernel contractions by showing how to define incision functions from selection functions and the other way round. We showed under which conditions going back and forth in this way yields identical functions. The properties that are needed to guarantee this are *completeness* for selection functions, and *relevance* for incision functions. We proved that *relevance* implies smoothness but not vice versa; in the special case when the belief base is a belief set, both properties for incision functions coincide.

REFERENCES

- [1] Carlos Alchourrón, Peter Gärdenfors, and David Makinson, ‘On the Logic of Theory Change: Partial Meet Contraction and Revision Functions’, *The Journal of Symbolic Logic*, **50**, 510–530, (1985).
- [2] Carlos Alchourrón and David Makinson, ‘On the Logic of Theory Change: Contraction Functions and their Associated Revision Functions’, *Theoria*, **48**, 14–37, (1982).
- [3] Johan de Kleer, ‘An Assumption-based TMS’, *Artificial Intelligence*, **28**, 127–162, (1986).
- [4] Simon Dixon and Norman Foo, ‘Connections Between the ATMS and AGM Belief Revision’, in *Proceedings of IJCAI-93*, pp. 534–539, (1993).
- [5] Jon Doyle, ‘A truth maintenance system’, *Artificial Intelligence*, **12**, 231–272, (1979).
- [6] Marcelo A. Falappa, Gabriele Kern-Isenberner, and Guillermo R. Simari, ‘Belief Revision, Explanations and Defeasible Reasoning’, *Artificial Intelligence Journal*, **141**, 1–28, (2002).
- [7] Eduardo Fermé, Karina Saez, and Pablo Sanz, ‘Multiple Kernel Contraction’, *Studia Logica*, **73**(2), 183–195, (2003).
- [8] Sven Ove Hansson, ‘Kernel Contraction’, *The Journal of Symbolic Logic*, **59**, 845–859, (1994).
- [9] Sven Ove Hansson, ‘Semi-Revision’, *Journal of Applied Non-Classical Logic*, **7**, 151–175, (1997).
- [10] Sven Ove Hansson, *A Textbook of Belief Dynamics: Theory Change and Database Updating*, Kluwer Academic Publishers, 1999.
- [11] Guillermo R. Simari and Ron P. Loui, ‘A Mathematical Treatment of Defeasible Reasoning and its Implementation’, *Artificial Intelligence*, **53**, 125–157, (1992).

Representing Relative Direction as a Binary Relation of Oriented Points

Reinhard Moratz¹

Abstract. A central issue in robotics is the representation of relative orientation. Currently, the standard solution utilizes metrical representations. The main reason for this might be that representing relatively fine distinctions is useful in many robotics tasks. If qualitative spatial constraint calculi are to be applied to cognitive robotics, they therefore have to afford relatively fine distinctions. The challenge for us then is to find a calculus which allows these fine distinctions, and yet is still simple enough to provide a provably minimal composition table.

In this paper we introduce a new calculus about oriented points which has a scalable granularity. In this calculus, named \mathcal{OPRA} , simple rules can generate the minimal composition table. Furthermore, the algebraic closure for a set of \mathcal{OPRA} statements is sufficient to solve knowledge integration tasks in robotics.

Keywords: Qualitative Spatial Reasoning, Cognitive Robotics

1 INTRODUCTION

Qualitative Reasoning about space abstracts from the physical world and enables computers to make predictions about spatial relations, even when a precise quantitative information is not available [2]. Spatial reasoning is then a central problem in formalizing common-sense knowledge [4]. The two main trends in Qualitative Spatial Reasoning are topological reasoning about regions [2, 16] and positional reasoning about point configurations [5]. Positional reasoning, i.e. distance and orientation, in particular is important for robot navigation [12].

In this paper we present the Oriented Point Algebra \mathcal{OPRA}_m which has a scalable granularity with parameter $m \in \mathbb{N}$. The motivation for this scalable granularity is that representing relatively fine distinctions is useful in many robotics tasks.

The rest of the paper is organized as follows: we will introduce the \mathcal{OPRA}_m calculus. First we will give a definition for the coarsest type ($m = 1$), followed by the model for arbitrary $m \in \mathbb{N}$. Then we will present an algorithm to generate the composition table. In the end we demonstrate the use of the new calculus for robot applications.

2 THE ORIENTED POINT ALGEBRA

Objects and locations are represented as simple, featureless points in aforementioned approaches on orientations. In contrast, our paper

presents a positional calculus which uses more complex basic entities: It is based on objects which are represented as oriented points. It is related to a calculus which is based on straight line segments (dipoles) [11]. Conceptually, the new calculus can be viewed as a transition from oriented line segments with concrete length to line segments with infinitely small length. In this conceptualization the length of the objects no longer has any importance. Thus, only the direction of the objects is modeled. *O-points*, our term for oriented points, are specified as pair of a point and a direction on the 2D-plane.

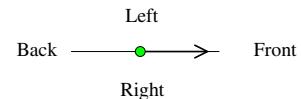


Figure 1. An oriented point and its qualitative spatial relative directions

2.1 Reasoning with Coarse O-Point Relations

In the coarsest representation a single o-point induces the sectors depicted in figure 1. “Front” and “Back” are linear sectors. “Left” and “Right” are half-planes. The position of the point itself is denoted as “Same”. A qualitative spatial relative orientation relation between two o-points is represented by the sector in which the second o-point lies with respect to the first one and by the sector in which the first one lies with respect to the second one.

For the general case of the two points having different positions we use the concatenated string of both sector names as the relation symbol. Then the configuration shown in figure 2 is expressed with the relation A RightLeft B . If both points share the same position the relation symbol starts with the word “Same” and the second substring denotes the direction of the second o-point with respect to the first one as shown in figure 3.

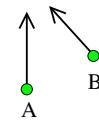


Figure 2. Qualitative spatial relation between two oriented points at different positions. The qualitative spatial relation depicted here is A RightLeft B .

Altogether we obtain 20 different atomic relations (four times four general relations plus four with the o-points at the same position). These relations are jointly exhaustive and pairwise disjoint (JEPD). The relation SameFront is the identity relation. We use \mathcal{OP}_1 to refer to the set of 20 atomic relations, and \mathcal{OPRA}_1 to refer to the power set of \mathcal{OP}_1 which contains all 2^{20} possible unions of the atomic relations.

¹ Transregional Collaborative Research Center “Spatial Cognition”, Faculty of Mathematics and Informatics, University of Bremen, moratz@informatik.uni-bremen.de

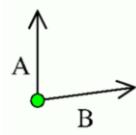


Figure 3. Qualitative spatial relation between two oriented points located at the same position. The qualitative spatial relation depicted here is $A \text{ SameRight } B$.

For reasoning about the o-point relations we apply constraint-based reasoning techniques which were originally introduced for temporal reasoning [1] and also proved valuable for spatial reasoning [16, 6]. In order to apply these techniques to a set of relations, the relations ideally should form a relation algebra [7], i.e. the atomic relations must be jointly exhaustive and pairwise disjoint and they must be closed under composition (\circ), intersection (\cap), complement (\neg), and converse (\sim). There must also be an empty relation, a universal relation (\top), and an identity relation. However, a new result from Renz and Ligozat [14] showed that the composition operation need not to be closed in order to successfully apply constraint-based reasoning techniques²

While the converse, the complement, and the intersection of relations can be computed from the set-theoretic definitions of the relations, the composition of relations must be computed based on the semantics of the relations. The compositions are usually computed only for the atomic relations and then stored in a composition table. The composition of compound relations can be obtained as the union of the compositions of the corresponding atomic relations. The compositions of the atomic relations can be deduced directly from the geometric semantics of the relations (see section 2.3).

O-point constraints are written as xRy where x, y are variables for o-points and R is a \mathcal{OPRA}_1 relation. Given a set Θ of o-point constraints, an important reasoning problem is deciding whether Θ is *consistent*, i.e., whether there is an assignment of all variables of Θ with dipoles such that all constraints are satisfied (a *solution*). We call this problem OPSAT. OPSAT is a Constraint Satisfaction Problem (CSP) [8] and can be solved using the standard methods developed for CSPs with infinite domains (see, e.g., [7]).

A partial method for determining inconsistency of a set of constraints Θ is the *path-consistency method* [8], which computes the algebraic closure on Θ . This method applies the following operation until a fixed point is reached:

$$\forall i, j, k : R_{ij} \leftarrow R_{ij} \cap (R_{ik} \circ R_{kj})$$

where i, j, k are nodes and R_{ij} is the relation between i and j . The resulting set of constraints is equivalent to the original set, i.e. it has the same set of solutions. If the empty relation occurs while performing this operation, Θ is inconsistent, otherwise the resulting set is path-consistent provided that the algebra has a strong composition. It was recently shown by Renz and Ligozat [14] that this algebraic closure procedure is also adequate for calculi with weak composition operation.

\mathcal{OPRA}_1 is very expressive already. Problems of pseudoline arrangements [17] can be expressed in \mathcal{OPRA}_1 . Since the stetchability of pseudoline problem is NP-hard [17], general OPSAT problems

² In section 2.3 we show how to generate a minimal composition table for \mathcal{OPRA} which guarantees at least a *weak* composition which is defined as minimal but not necessarily closed composition and is the substitute for the strong composition of a full relation algebra. Whether the \mathcal{OPRA} calculus is closed under composition is still an open question.

are NP-hard. However, sets of \mathcal{OPRA} relations can be expressed as equalities over polynomials with integer coefficients (which also holds for the finer grained o-point calculi introduced in the next section). Systems of such equalities can be solved using polynomial space [13].

2.2 Finer Grained O-Point Calculi

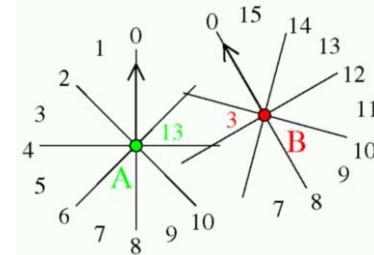


Figure 4. Two o-points in relation $A \text{ } 4\angle^3_{13} \text{ } B$

The design principle for \mathcal{OPRA}_1 can be generalized to calculi \mathcal{OPRA}_m with arbitrary $m \in \mathbb{N}$. Then an angular resolution of $\frac{2\pi}{4m}$ is used for the representation (a similar scheme for absolute direction instead of relative direction was recently designed by Renz and Mitra [15]).

To formally specify the o-point relations we use two-dimensional continuous space, in particular \mathbb{R}^2 . Every o-point S on the plane is an ordered pair of a point \mathbf{p}_S represented by its Cartesian coordinates x and y , with $x, y \in \mathbb{R}$ and a direction ϕ_S .

$$S = (\mathbf{p}_S, \phi_S), \quad \mathbf{p}_S = ((\mathbf{p}_S)_x, (\mathbf{p}_S)_y)$$

We distinguish the relative locations and orientations of the two o-points A and B expressed by a calculus \mathcal{OPRA}_m according to the following scheme. We use the symbol φ_{AB} for $\tan^{-1} \frac{(\mathbf{p}_B)_y - (\mathbf{p}_A)_y}{(\mathbf{p}_B)_x - (\mathbf{p}_A)_x}$ (\tan^{-1} has two arguments, the numerator, and the denominator, and maps to the interval $[0, 2\pi]$). According to the cyclic order of the directions it is appropriate to enumerate them by using the $4m$ elements of the cyclic group \mathcal{Z}_{4m} .

If $\mathbf{p}_A \neq \mathbf{p}_B$ the relation $A \text{ } m\angle_i^j \text{ } B$ ($i, j \in \mathcal{Z}_{4m}$) reads like this: Given a granularity m , the relative position of B with respect to A is described by j and the relative position of A with respect to B is described by i . Formally, it represents the following set of configurations:

$$\begin{aligned} & (((i \equiv_2 1) \wedge (2\pi \frac{i-1}{4m} < \varphi_{AB} - \phi_A < 2\pi \frac{i+1}{4m})) \\ & \vee ((i \equiv_2 0) \wedge (\varphi_{AB} - \phi_A = 2\pi \frac{i}{4m}))) \\ & \wedge (((j \equiv_2 1) \wedge (2\pi \frac{j-1}{4m} < \varphi_{AB} - \phi_B < 2\pi \frac{j+1}{4m})) \\ & \vee ((j \equiv_2 0) \wedge (\varphi_{AB} - \phi_B = 2\pi \frac{j}{4m}))) \end{aligned}$$

$a \equiv_2 b$ stands for $a \bmod 2 = b$. Figure 4 shows the resulting granularity for $m = 4$. Using this notation, a simple manipulation of the parameters yields the converse operation $(_m\angle_j^i)^\sim = {}_m\angle_i^j$.

If $\mathbf{p}_A = \mathbf{p}_B$, the relation $A \text{ } m\angle_i \text{ } B$ represents the following set of configurations:

$$\begin{aligned} & ((i \equiv_2 0) \wedge (\phi_B - \phi_A = 2\pi \frac{i}{4m})) \\ & \vee ((i \equiv_2 1) \wedge (2\pi \frac{i-1}{4m} < \phi_B - \phi_A < 2\pi \frac{i+1}{4m})) \end{aligned}$$

Hence the relation for two identical o-points $A = B$ for arbitrary $m \in \mathbb{N}$ is $A_m \angle 0B$. Using this notation a simple manipulation of the parameters yields the converse operation $(_m\angle_i)^\sim = {}_m\angle(4m - i)$.

The composition tables for the atomic relations of the OPRA_m calculi can be generated using a very simple enumeration strategy sketched in the following subsection.

In situations where the orientation of the second o-point is unknown, underspecified, or unimportant the notation $A \text{ } m\angle_i^? B$ expresses the information about the position of the second o-point without restricting the position of the first o-point with respect to the second one (e.g. the symbol $?$ denotes an index which can range over the whole set \mathcal{Z}_{4m}).

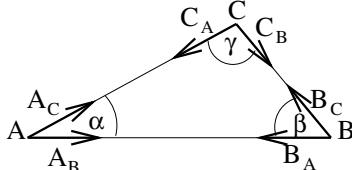


Figure 5. The sum of angles in a triangle equals π

2.3 Generating the composition table

The composition table can be viewed as a list (set) of all relation triples $Ar_{ab}B, Br_{bc}C, Cr_{ca}A$ for which r_{ab}, r_{bc} , and r_{ca} are consistent (A, B , and C being arbitrary o-points on the \mathbb{R}^2 plane). We give a very simple two-step procedure how to enumerate all these valid triples. Then we show that the list is complete.

In order to generate the o-point triple list we first look at general features of triangles on the 2D plane. With A_B we denote the o-point positioned at A and oriented towards position B . In the following, i , j , and k , and the according arithmetic operators are still defined in \mathcal{Z}_{4m} . For a triangle $(\alpha + \beta + \gamma = \pi)$ defined by the o-points A_B, B_C and C_A with $A_B \text{ } m\angle_i^? C_A, B_C \text{ } m\angle_j^? A_B$, and $C_A \text{ } m\angle_k^? B_C$ it holds that

$$2m - 1 \leq i + j + k \leq 2m + 1$$

The set of all possible qualitative triangle descriptions following the above scheme is denoted by T_m (which is enumerated by Algorithm 1).

$$T_m = \left\{ t_{(i,j,k)} \mid A_B \text{ } m\angle_i^? C_A \wedge B_C \text{ } m\angle_j^? A_B \wedge C_A \text{ } m\angle_k^? B_C \right\}$$

Now we can take each triangle description and generate the corresponding triples of OPRA_m relations. As a result we enumerated all triples of OPRA_m relations. But we might generate a specific triple several times which is no problem for our investigation and for the intended applications. Algorithm 2 (see appendix) gets a qualitative triangle (description) as input and generates all corresponding triples of OPRA_m relations. A consistent set of three OPRA_m relations $A \text{ } m\angle_{q_{ab}}^? B, B \text{ } m\angle_{q_{bc}}^? C$, and $C \text{ } m\angle_{q_{ca}}^? A$ can be represented as triple of pairs of the corresponding indices $\langle (q_{ab}, q_{ba}), (q_{bc}, q_{cb}), (q_{ca}, q_{ac}) \rangle$. Then algorithm 2 generates the set

$$\begin{aligned} C_{mijk} = & \left\{ \langle (q_{ab}^l, q_{ba}^l), (q_{bc}^l, q_{cb}^l), (q_{ca}^l, q_{ac}^l) \rangle \mid \right. \\ & A_B \text{ } m\angle_i^? C_A \wedge B_C \text{ } m\angle_j^? A_B \wedge C_A \text{ } m\angle_k^? B_C \quad \wedge \\ & \left. A \text{ } m\angle_{q_{ab}^l}^? B \wedge B \text{ } m\angle_{q_{bc}^l}^? C \wedge C \text{ } m\angle_{q_{ca}^l}^? A \right\} \end{aligned}$$

Note that A and A_B , B and B_C , C and C_A , respectively, have the same position on the \mathbb{R}^2 plane but not necessarily the same direction.

Now the composition table is generated by running algorithm 2 for every entry of the list generated by algorithm 1 and then deleting

Algorithm 1 Enumerating all qualitative triangle descriptions of granularity m

```

procedure EnumerateTriangleDescriptions( $\langle T_m, m \rangle$ )
1:  $T_m \leftarrow \emptyset$ 
2: for all  $0 \leq i \leq 2m$  do
3:   for all  $2m - i \leq j \leq 2m$  do
4:     if  $i \equiv_2 0 \wedge j \equiv_2 0$  then
5:        $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j)}\}$ 
6:     end if
7:     if  $i \equiv_2 0 \wedge j \equiv_2 1$  then
8:        $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j+1)}\}$ 
9:     end if
10:    if  $i \equiv_2 1 \wedge j \equiv_2 0$  then
11:       $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j+1)}\}$ 
12:    end if
13:    if  $i \equiv_2 1 \wedge j \equiv_2 1$  then
14:      if  $2m - i - j - 1 > 0$  then
15:         $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j-1)}\}$ 
16:      end if
17:      if  $2m - i - j > 0$  then
18:         $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j)}\}$ 
19:      end if
20:       $T_m \leftarrow T_m \cup \{t_{(i,j,2m-i-j+1)}\}$ 
21:    end if
22:  end for
23: end for

```

multiple copies of the same entry. Using this list the composition of two OPRA_m relations $(m\angle_{q_{ab}}^? \circ m\angle_{q_{bc}}^?)$ can be found by collecting all the entries $m\angle_{q_{ca}^1}^?, m\angle_{q_{ca}^2}^?, \dots$ which can be combined with the two operands and yield entries in the list of consistent relation triples produced by algorithms 1 and 2.

Theorem 1 All OPRA triples for granularity m are generated by algorithms 1 and 2.

Proof Sketch. Each OPRA triple corresponds to a qualitative triangle. A qualitative triangle is characterised by its three angles. The first angle can be any of the $2m + 1$ convex angles, the second has to be chosen from the set of convex angles which close the triangle. Then there are at most three possibilities for the third angle left.

All combinations of relative position and relative orientation are found now. The absolute orientations of the three o-points now can be enumerated independently. \square

Theorem 2 Every OPRA triple generated by algorithms 1 and 2 has an associated realisation in the intended model, the \mathbb{R}^2 plane.

Proof Sketch. The qualitative triangle which corresponds to the OPRA triple can be constructed in a straightforward manner: First, one side is assigned length 1. In the next step both adjacent angles get permissible values with increment $\frac{1}{4} \cdot \frac{2\pi}{4m}$. The third angle necessarily has a correct assignment generated by the set of assignments for the first two angles. Also all constraints for the global orientations of the three o-points are satisfied by members of this small set. \square

There are methods under development to directly compute the OPRA composition [9]. The OPRA calculus can be used to express many other qualitative calculi [3].

3 APPLYING THE *OPRA* CALCULUS TO SPATIAL AGENTS

The composition table becomes fundamentally more compact if we focus on o-point relations which can be found between the six o-points of a triangle (see figure 5). For mappings between these relations we introduce the operation *reorientation* (with the symbol \curvearrowright):

$$C_B \angle_j^0 A_C \implies C_A ({}_m\angle_j^0) \curvearrowright B_C = C_A \angle_{-j}^0 B_C$$

We will now demonstrate how to integrate local knowledge into survey knowledge with the presented calculus. The context is a mobile robot able to perceive colors and to segment simple objects (see figure 6). Furthermore, the system is able to perform straight motion steps and turnings. Since the robot has no prior knowledge about the sizes of the objects and does not know whether the ground is perfectly flat, it cannot estimate their distances. Thus, the robot has only access to local orientation information relative to its position.



Figure 6. Speech commands to a robot in a knowledge integration scenario

Users interact with our system by verbally issuing simple requests to the system³. To demonstrate the capabilities of our system we give a detailed description how the system performs knowledge integration on a specific example (see figure 6⁴). Figure 7 shows a slightly different configuration from a bird's eye view, using icons for the objects. The perceived constraints are:

$$t1_d \quad {}_6\angle_{23}^0 \quad t2_{t1} \quad (1)$$

$$t1_d \quad {}_6\angle_{23}^? \quad C1 \quad (2)$$

$$t1_d \quad {}_6\angle_{23}^? \quad C2 \quad (3)$$

The task is to "move to the yellow cube behind the red disk", i.e. one of the relations $D_{t1} \angle_{[9,15]}^? C1$ or $D_{t1} \angle_{[9,15]}^? C2$ must hold (${}_m\angle_{[k,l]}^j = {}_m\angle_k^j \cup {}_m\angle_{k+1}^j \cup \dots \cup {}_m\angle_l^j$).

The knowledge represented with equations 2 and 3 does not help the robot to distinguish both cube positions with respect to the disk. So the robot must move to acquire additional knowledge. In our example the robot moves straight towards a position roughly in the center of the perceived scene (in figure 7 the starting position is marked with $t1$ and the second position is marked with $t2$). Note that the robot only made straight movements and therefore still knows the direction of

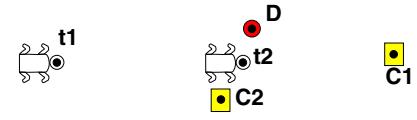


Figure 7. The two perspectives for the AIBO

its starting point. So the moving direction of the robot is still aligned with the robot's moving direction at the beginning. The corresponding directed line beginning with the starting point was also the justification for the knowledge represented in equation 1. The additional perceived knowledge at position $t2$ is:

$$t2_{t1} \quad {}_6\angle_{17}^0 \quad d_{t2} \quad (4)$$

$$t2_d \quad {}_6\angle_{19}^? \quad C1 \quad (5)$$

$$t2_d \quad {}_6\angle_{11}^? \quad C2 \quad (6)$$

Now the task is to use the perceived spatial knowledge for the decision which cube is in the referenced area. Constraint propagation until the algebraic closure is reached would be a solution (see section 2.1). Here we want to illustrate single reasoning steps and their effect in a demonstrative manner. We apply a composition on equations 1 and equation 4:

$$t1_d \quad {}_6\angle_{23}^0 \circ {}_6\angle_{17}^0 d_{t2} \implies t1_d \quad {}_6\angle_0^{[19,21]} d_{t2} \quad (7)$$

Then we apply a converse and a reorientation operation on equation 7:

$$d_{t1} \left(\left({}_6\angle_0^{[19,21]} \right) \curvearrowright \right) \curvearrowright t2_d \implies d_{t1} \quad {}_6\angle_{[3,5]}^0 t2_d \quad (8)$$

Now two compositions provide us further information (equation 8 composed with equation 5, equations 8, 6 respectively):

$$d_{t1} \quad {}_6\angle_{[3,5]}^0 \circ {}_6\angle_{19}^? C1 \implies d_{t1} \top C1 \quad (9)$$

$$d_{t1} \quad {}_6\angle_{[3,5]}^0 \circ {}_6\angle_{11}^? C2 \implies d_{t1} \quad {}_6\angle_{[1,5]}^? C2 \quad (10)$$

Equation 10 shows that cube 2 cannot be the goal cube. Equation 9 is the general relation, e.g. no information is produced. The relation between d_{t1} and $C1$ can be restricted better by composing the tautological constraint $d_{t1} \angle_0^0 t1_d$ with equation 2:

$$d_{t1} \angle_0^0 \circ {}_6\angle_{23}^? C1 \implies d_{t1} \angle_{[1,11]}^? C1 \quad (11)$$

As a result we could infer that cube 1 is the only cube which has a chance to be behind the disk.

For this scenario we generated ten random configurations and tested the reasoning. We only counted runs of the Aibo were no slipping of the legs distorted the straight movement. From the ten cases three could already be solved by local perception. Four cases made successful use of constraint propagation. Three cases were still ambiguous after constraint propagation. This means that in seven out of ten cases the chosen granularity was adequate in our scenario.

4 CONCLUSION

We presented a calculus for representing and reasoning about qualitative relative orientation information. Oriented points serve as the

³ for linguistic aspects of human robot interaction we refer to [10].

⁴ The distances on the image are smaller than in the real experiments for illustration purposes.

basic entities since they are the simplest spatial entities that have an intrinsic orientation. We identified systems of atomic relations on different granularity levels between o-points and identified a scheme for computing the calculi's composition tables based on their geometric semantics. We demonstrated on a small sample application how to represent spatial knowledge and perform reasoning in cognitive robotics with the new calculus.

Acknowledgment

The author would like to thank Lutz Frommberger, Frank Dylla, Marco Ragni, Jochen Renz, Diedrich Wolter, and Christian Freksa for interesting and helpful discussions related to the topic of the paper. The work was supported by the DFG Transregional Collaborative Research Center SFB/TR 8 "Spatial Cognition".

REFERENCES

- [1] J. F. Allen, 'Maintaining knowledge about temporal intervals', *Communications of the ACM*, 832–843, (1983).
- [2] Anthony G. Cohn, 'Qualitative spatial representation and reasoning techniques', in *KI-97: 21st Annual German Conference on Artificial Intelligence, Proceedings*, pp. 1–30. Springer, (1997).
- [3] Frank Dylla and Jan-Oliver Wallgrün, 'On generalizing orientation information in \mathcal{OPRA}_m ', in *KI-2006: Proceedings of the 29th Annual German Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence. Springer, (2006).
- [4] Kenneth D. Forbus, Paul Nielsen, and Boi Faltings, 'Qualitative spatial reasoning: The clock project', *Artificial Intelligence*, **51**(1–3), 417–471, (1991).
- [5] Christian Freksa, 'Using orientation information for qualitative spatial reasoning', in *Theories and methods of spatio-temporal reasoning in geographic space*, eds., A. U. Frank, I. Campari, and U. Formentini, 162–178, Springer, Berlin, (1992).
- [6] A. Islı and A.G. Cohn, 'A new approach to cyclic ordering of 2d orientations using ternary relation algebras', *Artificial Intelligence*, **122**(1–2), 137–187, (2000).
- [7] P. Ladkin and R. Maddux, 'On binary constraint problems', *Journal of the Association for Computing Machinery*, **41**(3), 435–469, (1994).
- [8] A. K Mackworth, 'Consistency in networks of relations', *Artificial Intelligence*, **8**, 99–118, (1977).
- [9] R. Moratz, F. Dylla, and L. Frommberger, 'A relative orientation algebra with adjustable granularity', in *IJCAI 2005 Workshop on Cognitive Robotics*, (2005).
- [10] R. Moratz and T. Tenbrink, 'Spatial reference in linguistic human-robot interaction: Iterative, empirically supported development of a model of projective relations', *Spatial Cognition and Computation*, **6**(1), 63–107, (2006).
- [11] Reinhard Moratz, Jochen Renz, and Diedrich Wolter, 'Qualitative spatial reasoning about line segments', in *Proceedings of ECAI 2000*, pp. 234–238, (2000).
- [12] A. Musto, K. Stein, A. Eisenkolb, and T. Röfer, 'Qualitative and quantitative representations of locomotion and their application in robot navigation', in *Proceedings IJCAI-99*, pp. 1067–1072, (1999).
- [13] J. Renegar, 'On the computational complexity and geometry of the first order theory of the reals. part i-iii', *Journal of Symbolic Computation*, **13**:3, 255–352, (1992).
- [14] J. Renz and G. Ligozat, 'Weak composition for qualitative spatial and temporal reasoning', in *CP 2005*, (2005).
- [15] J. Renz and D. Mitra, 'Qualitative direction calculi with arbitrary granularity', in *Proceedings PRICAI 2004 (LNAI 3157)*, (September 2004).
- [16] J. Renz and B. Nebel, 'Efficient methods for qualitative spatial reasoning', in *Proceedings ECAI-98*, pp. 562–566, Brighton, (1998).
- [17] P. Shor, 'Strechability of pseudolines is np-hard', in *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, eds., P. Gritzmann and B. Sturmfels, 531–554, AMS-ACM, (1991).

Algorithm 2 Generating all OPRA triples for a given triangle

```

procedure generateOPRAtriples( $\langle t_{\langle i,j,k \rangle}, m \rangle$ )
1:  $C_{mijk} \leftarrow \emptyset$ 
2: for all  $0 \leq o_{ab} \leq 4m$  do
3:    $q_{ab} \leftarrow o_{ab}$ 
4:    $q_{ac} \leftarrow o_{ab} + i$ 
5:   generate2ndOpointVariations( $q_{ab}, q_{ac}, C_{mijk}$ )
6:   if  $i \equiv_2 1 \wedge o_{ab} \equiv_2 1$  then
7:      $q_{ac} \leftarrow o_{ab} + i - 1$ 
8:     generate2ndOpointVariations( $q_{ab}, q_{ac}, C_{mijk}$ )
9:      $q_{ac} \leftarrow o_{ab} + i + 1$ 
10:    generate2ndOpointVariations( $q_{ab}, q_{ac}, C_{mijk}$ )
11:   end if
12: end for

procedure generate2ndOpointVariations( $q_{ab}, q_{ac}, C_{mijk}$ )
1: for all  $0 \leq o_{bc} \leq 4m$  do
2:    $q_{bc} \leftarrow o_{bc}$ 
3:    $q_{ba} \leftarrow o_{bc} + j$ 
4:   generate3rdOpointVariations( $q_{ab}, q_{ac}, q_{bc}, q_{ba}, C_{mijk}$ )
5:   if  $j \equiv_2 1 \wedge o_{bc} \equiv_2 1$  then
6:      $q_{ba} \leftarrow o_{bc} + j - 1$ 
7:     generate3rdOpointVariations( $q_{ab}, q_{ac}, q_{bc}, q_{ba}, C_{mijk}$ )
8:      $q_{ba} \leftarrow o_{bc} + j + 1$ 
9:     generate3rdOpointVariations( $q_{ab}, q_{ac}, q_{bc}, q_{ba}, C_{mijk}$ )
10:   end if
11: end for

procedure generate3rdOpointVariations( $q_{ab}, q_{ac}, q_{bc}, q_{ba}, C_{mijk}$ )
1: for all  $0 \leq o_{ca} \leq 4m$  do
2:    $q_{ca} \leftarrow o_{ca}$ 
3:    $q_{cb} \leftarrow o_{ca} + k$ 
4:    $C_{mijk} \leftarrow C_{mijk} \cup \{(q_{ab}, q_{ba}), (q_{bc}, q_{cb}), (q_{ca}, q_{ac})\}$ 
5:   if  $k \equiv_2 1 \wedge o_{ca} \equiv_2 1$  then
6:      $q_{cb} \leftarrow o_{ca} + k - 1$ 
7:      $C_{mijk} \leftarrow C_{mijk} \cup \{(q_{ab}, q_{ba}), (q_{bc}, q_{cb}), (q_{ca}, q_{ac})\}$ 
8:      $q_{cb} \leftarrow o_{ca} + k + 1$ 
9:      $C_{mijk} \leftarrow C_{mijk} \cup \{(q_{ab}, q_{ba}), (q_{bc}, q_{cb}), (q_{ca}, q_{ac})\}$ 
10:   end if
11: end for

```

Modular Equivalence for Normal Logic Programs

Emilia Oikarinen and Tomi Janhunen¹

Abstract. A Gaifman-Shapiro-style architecture of program modules is introduced in the case of normal logic programs under stable model semantics. The composition of program modules is suitably limited by module conditions which ensure the compatibility of the module system with stable models. The resulting module theorem properly strengthens Lifschitz and Turner's splitting set theorem [17] for normal logic programs. Consequently, the respective notion of equivalence between modules, i.e. modular equivalence, proves to be a congruence relation. Moreover, it is shown how our translation-based verification method [15] is accommodated to the case of modular equivalence; and how the verification of weak/visible equivalence can be optimized as a sequence of module-level tests.

1 INTRODUCTION

Answer set programming (ASP) is a promising constraint programming paradigm [20, 19, 11] in which problems are solved by capturing their solutions as *answer sets* or *stable models* of logic programs. The development and optimization of logic programs in ASP gives rise to a meta-level problem of verifying whether subsequent programs are equivalent. To solve this problem, a translation-based approach has been proposed and extended further [14, 23, 21, 25]. The underlying idea is to combine two logic programs P and Q under consideration into two logic programs $\text{EQT}(P, Q)$ and $\text{EQT}(Q, P)$ which have no stable models iff P and Q are *weakly equivalent*, i.e. have the same stable models. This enables the use of the same ASP solver, such as SMODELS or DLV, for the equivalence verification problem as for the search of stable models in general. First experimental results [14, 21] suggest that the translation-based method can be effective and sometimes much faster than a simple cross-check.

As a potential limitation, the translation-based method as described above treats programs as integral entities and therefore no computational advantage is sought by breaking programs into smaller parts, say *modules* of some kind. Such an optimization strategy is largely preempted by the fact that weak equivalence, denoted by \equiv , fails to be a *congruence relation* for \cup , i.e. weak equivalence is not preserved under substitutions in unions of programs. More formally put, $P \equiv Q$ does not imply $P \cup R \equiv Q \cup R$ in general. The same can be stated about *uniform equivalence* [22] but not about *strong equivalence* [16] which admits substitutions by definition.

From our point of view, strong equivalence seems inappropriate for *fully modularizing* the verification task of weak equivalence because programs P and Q may be weakly equivalent even if they build on respective modules $P_i \subseteq P$ and $Q_i \subseteq Q$ that are not strongly equivalent. For the same reason, program transformations that are known to preserve strong equivalence [4] do not provide an inclusive basis for reasoning about weak equivalence. Nevertheless, there are

cases where one can utilize the fact that strong equivalence implies weak equivalence. For instance, if P and Q are composed of strongly equivalent pairs of modules P_i and Q_i for all i , then P and Q can be directly inferred to be strongly/weakly equivalent. These observations about strong equivalence motivate the strive for a weaker congruence relation compatible with weak equivalence at program-level.

To address the lack of a suitable congruence relation in the context of ASP, we propose a new design in this article. The design superficially resembles that of Gaifman and Shapiro [10] but stable model semantics [12] and special module conditions are incorporated. The feasibility of the design is crystallized in a *module theorem* which shows the module system fully compatible with stable models. In fact, the module theorem established here is a proper strengthening of the splitting set theorem established by Lifschitz and Turner [17] in the case of normal logic programs. The main difference is that our result allows negative recursion between modules. Moreover, it enables the introduction of a notion of equivalence, i.e. *modular equivalence*, which turns out to be a proper congruence relation and reduces to weak equivalence for program modules which have a completely specified input and no hidden (auxiliary) atoms. Such modules correspond to normal logic programs without auxiliary atoms. If normal programs P and Q are composed of modularly equivalent modules P_i and Q_i for all i , then P and Q are modularly equivalent, or equivalently stated, weakly equivalent. The notion of modular equivalence opens immediately new prospects as regards the translation-based verification method [14, 21]. First, the method can be tuned for the task of verifying modular equivalence by attaching a *context generator* to program modules in analogy to [25]. Second, we demonstrate how the verification of weak equivalence can be reorganized as a sequence of tests, each of which concentrates on a pair of respective modules in the programs subject to the verification task.

The rest of this article is structured as follows. As a preparatory step, the syntax and semantics of normal logic programs is recalled in Section 2. A summary of equivalence relations follows in Section 3. Section 4 concentrates on specifying program modules as well as establishing the module theorem discussed above. The notion of modular equivalence is then presented in Section 5 which also includes a proof of the congruence property and a brief account of computational complexity. Moreover, connections between modular equivalence and the translation-based method for verifying weak equivalence [14] are worked out. Section 6 briefly contrasts our work with earlier approaches. Finally, a conclusion is given in Section 7.

2 NORMAL LOGIC PROGRAMS

We present *normal logic programs* in the *propositional* case.

Definition 1 A *normal logic program (NLP)* is a finite set of rules of the form $h \leftarrow B^+, \sim B^-$, where h is an atom, B^+ and B^- are sets of atoms, and $\sim B = \{\sim b \mid b \in B\}$ for any set of atoms B .

¹ Helsinki University of Technology (TKK), P.O. Box 5400, FI-02015 TKK, Finland. Email:{Emilia.Oikarinen,Tomi.Janhunen}@tkk.fi

The symbol “ \sim ” denotes *default negation* and we define *default literals* as atoms a or their default negations $\sim a$. A rule consists of a *head* h and a *body* $B^+ \cup \sim B^-$. A rule is a *fact* whenever its body is empty, and “ \leftarrow ” is omitted. A rule is *positive*, if $B^- = \emptyset$. A NLP consisting of positive rules only is a *positive logic program*.

The *Herbrand base* $\text{Hb}(P)$ of a NLP P is any fixed set of atoms containing all atoms appearing in the rules of P . Moreover, the base $\text{Hb}(P)$ is supposed to be finite whenever P is. Under this definition, the Herbrand base of P may contain atoms which have no occurrences in P . This aspect is useful e.g. when programs are optimized and there is a need to keep track of removed atoms. Given a NLP P , an *interpretation* M of P is a subset of $\text{Hb}(P)$ defining which atoms $a \in \text{Hb}(P)$ are true ($a \in M$) and which are false ($a \notin M$). An interpretation $M \subseteq \text{Hb}(P)$ is a (*classical*) *model* of P , denoted by $M \models P$ iff $B^+ \subseteq M$ and $B^- \cap M = \emptyset$ imply $h \in M$ for each rule $h \leftarrow B^+, \sim B^- \in P$. For a positive program P , an interpretation $M \subseteq \text{Hb}(P)$ is the (unique) *least model* of P , denoted by $\text{LM}(P)$, iff there is no $M' \models P$ such that $M' \subset M$. *Stable models* as proposed by Gelfond and Lifschitz [12] generalize least models for normal logic programs.

Definition 2 An interpretation $M \subseteq \text{Hb}(P)$ is a *stable model* of a NLP P iff $M = \text{LM}(P^M)$ where the Gelfond-Lifschitz reduct $P^M = \{h \leftarrow B^+ \mid h \leftarrow B^+, \sim B^- \in P \text{ and } M \cap B^- = \emptyset\}$.

Stable models are not necessarily unique in general: a NLP may have several stable models or no stable models at all. The set of stable models of a NLP P is denoted by $\text{SM}(P)$.

The *positive dependency relation* $\leq \subseteq \text{Hb}(P) \times \text{Hb}(P)$ of P is the reflexive and transitive closure of a relation \leq_1 defined as follows. Given $a, b \in \text{Hb}(P)$, we say that b depends directly on a , denoted $a \leq_1 b$, iff there is a rule $b \leftarrow B^+, \sim B^- \in P$ such that $a \in B^+$. The *positive dependency graph* of P , denoted by $\text{Dep}^+(P)$, is a graph with $\text{Hb}(P)$ and $\{\langle b, a \rangle \mid a \leq_1 b\}$ as the sets of vertices and edges, respectively. A *strongly connected component* of $\text{Dep}^+(P)$ is a maximal subset $C \subseteq \text{Hb}(P)$ such that $a \leq b$ holds for all $a, b \in C$. The strongly connected components of $\text{Dep}^+(P)$ partition $\text{Hb}(P)$ into equivalence classes. The dependency relation \leq can then be generalized for strongly connected components: $C_i \leq C_j$, i.e. C_j depends on C_i , iff $c_i \leq c_j$ for any $c_i \in C_i$ and any $c_j \in C_j$.

A *splitting set* for a NLP P [17] is any set $U \subseteq \text{Hb}(P)$ such that for every $h \leftarrow B^+, \sim B^- \in P$, if $h \in U$ then $B^+ \cup B^- \subseteq U$. The *bottom* of P relative to U is $b_U(P) = \{h \leftarrow B^+, \sim B^- \in P \mid \{h\} \cup B^+ \cup B^- \subseteq U\}$, and the *top* of P relative to U is $t_U(P) = P \setminus b_U(P)$. The top can be partially evaluated with respect to an interpretation $X \subseteq U$. The result is a program $e(t_U(P), X)$ that contains a rule $h \leftarrow (B^+ \setminus U), \sim(B^- \setminus U)$ for each $h \leftarrow B^+, \sim B^- \in t_U(P)$ such that $B^+ \cap U \subseteq X$ and $(B^- \cap U) \cap X = \emptyset$. Given a splitting set U for a NLP P , a *solution* to P with respect to U is a pair $\langle X, Y \rangle$ such that $X \subseteq U$, $Y \subseteq \text{Hb}(P) \setminus U$, $X \in \text{SM}(b_U(P))$, and $Y \in \text{SM}(e(t_U(P), X))$. The *splitting set theorem* by Lifschitz and Turner [17] relates solutions with stable models. Given a NLP P , a splitting set U for P , and $M \subseteq \text{Hb}(P)$, it holds that $M \in \text{SM}(P)$ iff $\langle M \cap U, M \setminus U \rangle$ is a solution to P with respect to U .

3 NOTIONS OF EQUIVALENCE

Next we briefly review a number of equivalence relations for NLPs. As stated in Section 1, two NLPs P and Q are (weakly) equivalent, denoted by $P \equiv Q$, iff $\text{SM}(P) = \text{SM}(Q)$. Lifschitz et al. [16] introduce a much stronger notion: P and Q are *strongly equivalent*,

denoted $P \equiv_s Q$, iff $P \cup R \equiv Q \cup R$ for any NLP R acting as a context. Yet another relation originates from the database community [22]: P and Q are *uniformly equivalent*, denoted $P \equiv_u Q$, iff $P \cup F \equiv Q \cup F$ for any set of facts F . It is clear that $P \equiv_s Q$ implies $P \equiv_u Q$ which implies $P \equiv Q$, but not vice versa (in both cases).

Strongly equivalent logic programs are semantics preserving substitutes of each other and thus \equiv_s can be understood as a *congruence relation* among NLPs, i.e. if $P \equiv_s Q$, then $P \cup R \equiv_s Q \cup R$ for all NLPs R . On the other hand, \equiv_u is not a congruence for \cup , as shown in Example 1 below and the same applies to \equiv . Thus \equiv and \equiv_u are best suited for program-level rather than module-level comparisons.

Example 1 [4, Example 1] Consider NLPs $P = \{a.\}$ and $Q = \{a \leftarrow \sim b. a \leftarrow b.\}$. It holds $P \equiv_u Q$, but $P \cup R \not\equiv Q \cup R$ for $R = \{b \leftarrow a.\}$. Thus $P \not\equiv_s Q$ and \equiv_u is not a congruence for \cup .

For $P \equiv Q$ to hold, the stable models in $\text{SM}(P)$ and $\text{SM}(Q)$ have to be identical subsets of $\text{Hb}(P)$ and $\text{Hb}(Q)$, respectively, and the same can be stated about \equiv_s and \equiv_u . This makes these notions of equivalence less useful if $\text{Hb}(P)$ and $\text{Hb}(Q)$ differ by some atoms which formalize some auxiliary concepts and are not trivially false. Following the ideas from [13] we partition $\text{Hb}(P)$ into its *visible* and *hidden* parts $\text{Hb}_v(P)$ and $\text{Hb}_h(P)$, respectively. The idea behind *visible equivalence* is that atoms in $\text{Hb}_h(P)$ and $\text{Hb}_h(Q)$ are local to P and Q and negligible as regards the equivalence of P and Q .

Definition 3 [13] Normal logic programs P and Q are *visibly equivalent*, denoted by $P \equiv_v Q$, iff $\text{Hb}_v(P) = \text{Hb}_v(Q)$ and there is a bijection $f : \text{SM}(P) \rightarrow \text{SM}(Q)$ such that for all interpretations $M \in \text{SM}(P)$, $M \cap \text{Hb}_v(P) = f(M) \cap \text{Hb}_v(Q)$.

Note that the number of stable models is preserved under \equiv_v . Such a strict correspondence of models is much dictated by the ASP methodology: the stable models of a program usually correspond to the solutions of the problem being solved and thus the exact preservation of models is highly significant. In the fully visible case, i.e. $\text{Hb}_h(P) = \text{Hb}_h(Q) = \emptyset$, the relation \equiv_v becomes very close to \equiv . The only difference is the requirement $\text{Hb}(P) = \text{Hb}(Q)$ insisted by \equiv_v . This is of little importance as Herbrand bases can always be extended to meet $\text{Hb}(P) = \text{Hb}(Q)$. Since weak equivalence is not a congruence, visible equivalence cannot be a congruence either.

The *relativized variants* of \equiv_s and \equiv_u introduced by Woltran [25] allow the context to be constrained using a set of atoms A .

Definition 4 Two NLPs P and Q are *strongly equivalent relative to A* , denoted by $P \equiv_s^A Q$, iff $P \cup R \equiv Q \cup R$ for all NLPs R over the set of atoms A ; *uniformly equivalent relative to A* , denoted by $P \equiv_u^A Q$, iff $P \cup F \equiv Q \cup F$ for all sets of facts $F \subseteq A$.

Setting $A = \emptyset$ in the above reduces both $P \equiv_s^A Q$ and $P \equiv_u^A Q$ to \equiv . Thus neither of them is a congruence for \cup .

Eiter et al. [7] use *equivalence frames* to capture various kinds of equivalence relations such as those defined above. The relation \equiv_v makes an exception in this respect as it does not fit into equivalence frames based on *projected answer sets*. A projective variant of Definition 3 would require $\{M \cap \text{Hb}_v(P) \mid M \in \text{SM}(P)\} = \{N \cap \text{Hb}_v(Q) \mid N \in \text{SM}(Q)\}$. As a consequence, the number of answer sets might not be preserved which we find to contrast the general nature of ASP as discussed after Definition 3.

4 MODULAR LOGIC PROGRAMS

We define *logic program modules* similarly to [10] but consider the case of NLPs instead of positive (disjunctive) logic programs.

Definition 5 A triple $\mathbb{P} = (P, I, O)$ is a logic program module, if

1. P is a finite set of rules of the form $h \leftarrow B^+, \sim B^-$;
2. I and O are sets of propositional atoms such that $I \cap O = \emptyset$; and
3. $\text{Head}(P) \cap I = \emptyset$ where $\text{Head}(P) = \{h \mid h \leftarrow B^+, \sim B^- \in P\}$.

The Herbrand base of module \mathbb{P} , $\text{Hb}(\mathbb{P})$, is the set of atoms appearing in P combined with $I \cup O$. Intuitively the set I defines the *input* of a module and the set O is the *output*. The atoms in $I \cup O$ are visible, i.e. the visible Herbrand base of module \mathbb{P} is $\text{Hb}_v(\mathbb{P}) = I \cup O$. Notice that $I \cup O$ can also contain atoms not appearing in P similarly to the possibility of having additional atoms in the Herbrand bases of NLPs. All other atoms are hidden, i.e. $\text{Hb}_h(\mathbb{P}) = \text{Hb}(\mathbb{P}) \setminus \text{Hb}_v(\mathbb{P})$.

For the composition of modules we take the union of the disjoint sets of rules involved in them in analogy to [10]. The conditions given in [10] are not yet sufficient for our purposes, and we impose a further restriction denying positive recursion between modules.

Definition 6 Consider $\mathbb{P}_1 = (P_1, I_1, O_1)$ and $\mathbb{P}_2 = (P_2, I_2, O_2)$ and let C_1, \dots, C_n be the strongly connected components of $\text{Dep}^+(P_1 \cup P_2)$. There is a positive recursion between \mathbb{P}_1 and \mathbb{P}_2 , if $C_i \cap O_1 \neq \emptyset$ and $C_i \cap O_2 \neq \emptyset$ for some component C_i .

The idea is that all inter-module dependencies go through the input/output interface of the modules, and hidden atoms are local to each module. If there is a strongly connected component C_i in $\text{Dep}^+(P_1 \cup P_2)$ containing atoms from both O_1 and O_2 , we know that, some output atom a of \mathbb{P}_1 depends on some output atom b of \mathbb{P}_2 which again depends on a . This yields a positive recursion.

Definition 7 The join of modules $\mathbb{P}_1 = (P_1, I_1, O_1)$ and $\mathbb{P}_2 = (P_2, I_2, O_2)$, denoted by $\mathbb{P}_1 \sqcup \mathbb{P}_2$, is defined if

1. $O_1 \cap O_2 = \emptyset$;
2. $\text{Hb}_h(\mathbb{P}_1) \cap \text{Hb}(\mathbb{P}_2) = \text{Hb}_h(\mathbb{P}_2) \cap \text{Hb}(\mathbb{P}_1) = \emptyset$; and
3. there is no positive recursion between \mathbb{P}_1 and \mathbb{P}_2 .

Then $\mathbb{P}_1 \sqcup \mathbb{P}_2 = (P_1 \cup P_2, (I_1 \setminus O_2) \cup (I_2 \setminus O_1), O_1 \cup O_2)$.

Note that the first condition is implied by the third, and the second can be circumvented in practice by renaming the hidden atoms uniquely for each module. The join operation is *commutative* and *associative* whenever the respective joins are defined. The following hold for $\mathbb{P}_1 \sqcup \mathbb{P}_2$: $P_1 \cap P_2 = \emptyset$, $\text{Hb}_h(\mathbb{P}_1) \cap \text{Hb}_h(\mathbb{P}_2) = \emptyset$, and $\text{Hb}_v(\mathbb{P}_1) \cap \text{Hb}_v(\mathbb{P}_2) = (I_1 \cap I_2) \cup (I_1 \cap O_2) \cup (I_2 \cap O_1)$. Also, $\text{Hb}(\mathbb{P}_1 \sqcup \mathbb{P}_2) = \text{Hb}(\mathbb{P}_1) \cup \text{Hb}(\mathbb{P}_2)$, and similarly for the hidden and visible part of $\text{Hb}(\mathbb{P}_1 \sqcup \mathbb{P}_2)$, respectively. The conditions above impose no restrictions on positive dependencies *inside* modules or on *negative* dependencies in general. The input of $\mathbb{P}_1 \sqcup \mathbb{P}_2$ can be smaller than the union of inputs of individual modules because \mathbb{P}_1 may provide input for \mathbb{P}_2 , and conversely, as demonstrated below.

Example 2 The join of $\mathbb{P} = (\{a \leftarrow \sim b\}, \{b\}, \{a\})$ and $\mathbb{Q} = (\{b \leftarrow \sim a\}, \{a\}, \{b\})$ is $\mathbb{P} \sqcup \mathbb{Q} = (\{a \leftarrow \sim b. b \leftarrow \sim a\}, \emptyset, \{a, b\})$.

The stable semantics of a module is defined with respect to a given input, i.e. a subset of the input atoms of the module. Input is seen as a set of facts (or a database) to be added to the module.

Definition 8 The instantiation of a module $\mathbb{P} = (P, I, O)$ with an input $A \subseteq I$ is $\mathbb{P}(A) = \mathbb{P} \sqcup \mathbb{F}_A$, where $\mathbb{F}_A = (\{a. \mid a \in A\}, \emptyset, I)$.

Note that $\mathbb{P}(A) = (P \cup \{a. \mid a \in A\}, \emptyset, I \cup O)$ is essentially a NLP with $\text{Hb}_v(\mathbb{P}(A)) = I \cup O$, and we can generalize the stable model semantics for modules. In the sequel $\mathbb{P}(A)$ is identified with the respective NLP $P \cup F_A$, where $F_A = \{a. \mid a \in A\}$, and $M \cap I$ acts as a particular input with respect to which the module is instantiated.

Definition 9 An interpretation $M \subseteq \text{Hb}(\mathbb{P})$ is a stable model of $\mathbb{P} = (P, I, O)$, denoted $M \in \text{SM}(\mathbb{P})$, iff $M = \text{LM}(P^M \cup F_{M \cap I})$.

A concept of *compatibility* is used to describe when $M_1 \in \text{SM}(\mathbb{P}_1)$ can be combined with $M_2 \in \text{SM}(\mathbb{P}_2)$: stable models M_1 and M_2 are *compatible*, iff $M_1 \cap \text{Hb}_v(\mathbb{P}_2) = M_2 \cap \text{Hb}_v(\mathbb{P}_1)$. Theorem 1 relates program-level stability with module-level stability.

Theorem 1 (Module theorem). If $\mathbb{P}_1 \sqcup \mathbb{P}_2$ is defined for modules \mathbb{P}_1 and \mathbb{P}_2 , then $M \in \text{SM}(\mathbb{P}_1 \sqcup \mathbb{P}_2)$ iff $M_1 = M \cap \text{Hb}(\mathbb{P}_1) \in \text{SM}(\mathbb{P}_1)$, $M_2 = M \cap \text{Hb}(\mathbb{P}_2) \in \text{SM}(\mathbb{P}_2)$, and M_1 and M_2 are compatible.

Proof sketch. Suppose $\mathbb{P} = \mathbb{P}_1 \sqcup \mathbb{P}_2 = (P, I, O)$ is defined for modules $\mathbb{P}_1 = (P_1, I_1, O_1)$ and $\mathbb{P}_2 = (P_2, I_2, O_2)$. “ \Rightarrow ” Let $M \in \text{SM}(\mathbb{P})$. Then $M_1 = M \cap \text{Hb}(\mathbb{P}_1)$ and $M_2 = M \cap \text{Hb}(\mathbb{P}_2)$ are clearly compatible and it is straightforward to show that conditions 1 and 2 in Definition 7 imply $M_1 \in \text{SM}(\mathbb{P}_1)$ and $M_2 \in \text{SM}(\mathbb{P}_2)$. “ \Leftarrow ” Let $M_1 \in \text{SM}(\mathbb{P}_1)$, and $M_2 \in \text{SM}(\mathbb{P}_2)$ be compatible and define $M = M_1 \cup M_2$. There is a strict total ordering $<$ for the strongly connected components C_i of $\text{Dep}^+(P)$ such that if $C_i < C_j$, then $C_i \leq C_j$ and $C_j \not\leq C_i$; or $C_i \not\leq C_j$ and $C_j \not\leq C_i$. Let $C_1 < \dots < C_n$ be such an ordering. Show that exactly one of the following holds for each C_i : (i) $C_i \subseteq I$, (ii) $C_i \subseteq O_1 \cup \text{Hb}_h(\mathbb{P}_1)$, or (iii) $C_i \subseteq O_2 \cup \text{Hb}_h(\mathbb{P}_2)$. Finally, show by induction that $M \cap (\cup_{i=1}^k C_i) = \text{LM}(P^M \cup F_{M \cap I}) \cap (\cup_{i=1}^k C_i)$ for $0 \leq k \leq n$ by applying the splitting set theorem [17] to $P^M \cup F_{M \cap I}$. \square

Example 3 shows that condition 3 in Definition 7 is necessary to guarantee that local stability implies global stability.

Example 3 Consider $\mathbb{P}_1 = (\{a \leftarrow b.\}, \{b\}, \{a\})$ and $\mathbb{P}_2 = (\{b \leftarrow a.\}, \{a\}, \{b\})$ with $\text{SM}(\mathbb{P}_1) = \text{SM}(\mathbb{P}_2) = \{\emptyset, \{a, b\}\}$. The join of \mathbb{P}_1 and \mathbb{P}_2 is not defined because of positive recursion (conditions 1 and 2 in Definition 7 are satisfied, however). For a NLP $P = \{a \leftarrow b. b \leftarrow a.\}$, we get $\text{SM}(P) = \{\emptyset\}$. Thus, the positive dependency between a and b excludes $\{a, b\}$ from $\text{SM}(P)$.

Theorem 1 is strictly stronger than the splitting set theorem [17] for NLPs. If U is a splitting set for a NLP P , then $P = \mathbb{B} \sqcup \mathbb{T} = (b_U(P), \emptyset, U) \sqcup (t_U(P), U, \text{Hb}(P) \setminus U)$, and furthermore $M_1 \in \text{SM}(\mathbb{B})$ and $M_2 \in \text{SM}(\mathbb{T})$ iff $\langle M_1, M_2 \setminus U \rangle$ is a solution for P with respect to U . On the other hand the splitting set theorem cannot be applied to e.g. $\mathbb{P} \sqcup \mathbb{Q}$ from Example 2, since neither $\{a\}$ nor $\{b\}$ is a splitting set. Our theorem also strengthens a module theorem given in [13, Theorem 6.22] to cover NLPs involving positive body literals. Theorem 1 can easily be generalized for modules consisting of several submodules. Although Theorem 1 enables the computation of stable models on a module-by-module basis, it leaves us the task of excluding mutually incompatible combinations of stable models.

5 MODULAR EQUIVALENCE

The notion of *modular equivalence* combines features from relativized uniform equivalence and visible equivalence.

Definition 10 Logic program modules $\mathbb{P} = (P, I_P, O_P)$ and $\mathbb{Q} = (Q, I_Q, O_Q)$ are modularly equivalent, denoted by $\mathbb{P} \equiv_m \mathbb{Q}$, iff

1. $I_P = I_Q = I$ and $O_P = O_Q = O$, and
2. $\mathbb{P}(A) \equiv_v \mathbb{Q}(A)$ for all $A \subseteq I$.

Modular equivalence is very close to \equiv_v defined for modules. As a matter of fact, if Definition 3 is generalized for modules, the second condition in Definition 10 can be revised to $\mathbb{P} \equiv_v \mathbb{Q}$. That, however,

is not enough to cover the first condition in Definition 10, as \equiv only enforces $\text{Hb}_v(\mathbb{P}) = \text{Hb}_v(\mathbb{Q})$. If $I = \emptyset$, modular equivalence coincides with visible equivalence. If $O = \emptyset$, then $\mathbb{P} \equiv_m \mathbb{Q}$ means that \mathbb{P} and \mathbb{Q} have the same number of stable models on each input.

Furthermore, in the *fully visible case*, i.e. if $\text{Hb}_h(\mathbb{P}) = \text{Hb}_h(\mathbb{Q}) = \emptyset$, modular equivalence can be seen as a special case of A -uniform equivalence for $A = I$. Recall, however, the restrictions $\text{Head}(P) \cap I = \text{Head}(Q) \cap I = \emptyset$ imposed by module structure. With a further restriction $I = \emptyset$, modular equivalence coincides with weak equivalence. Setting $I = \text{Hb}(\mathbb{P})$ would give uniform equivalence, but the restriction $\text{Head}(P) \cap I = \emptyset$ leaves room for the empty module only.

Since \equiv_v is not a congruence relation for \sqcup , neither is modular equivalence. The situation changes, however, if one considers the join operation \sqcup which suitably restricts possible contexts.

Theorem 2 *Let \mathbb{P}, \mathbb{Q} and \mathbb{R} be logic program modules. If $\mathbb{P} \equiv_m \mathbb{Q}$ and both $\mathbb{P} \sqcup \mathbb{R}$ and $\mathbb{Q} \sqcup \mathbb{R}$ are defined, then $\mathbb{P} \sqcup \mathbb{R} \equiv_m \mathbb{Q} \sqcup \mathbb{R}$.*

Proof sketch. Consider $\mathbb{P} = (P, I, O)$ and $\mathbb{Q} = (Q, I, O)$ such that $\mathbb{P} \equiv_m \mathbb{Q}$, and $\mathbb{R} = (R, I_R, O_R)$ such that $\mathbb{P} \sqcup \mathbb{R}$ and $\mathbb{Q} \sqcup \mathbb{R}$ are defined. It follows from Theorem 1 that $M_P = M \cap \text{Hb}(\mathbb{P}) \in \text{SM}(\mathbb{P})$ and $M_R = M \cap \text{Hb}(\mathbb{R}) \in \text{SM}(\mathbb{R})$ for any $M \in \text{SM}(\mathbb{P} \sqcup \mathbb{R})$. Since $\mathbb{P} \equiv_m \mathbb{Q}$, there is a bijection $f : \text{SM}(\mathbb{P}) \rightarrow \text{SM}(\mathbb{Q})$ such that $M_P \cap (O \cup I) = f(M_P) \cap (O \cup I)$. Define $M_Q = f(M_P)$ and apply Theorem 1 to show that $M_Q \cup M_R \in \text{SM}(\mathbb{Q} \sqcup \mathbb{R})$. Finally show that $g : \text{SM}(\mathbb{P} \sqcup \mathbb{R}) \rightarrow \text{SM}(\mathbb{Q} \sqcup \mathbb{R})$ defined as $g(M) = f(M \cap \text{Hb}(\mathbb{P})) \cup (M \cap \text{Hb}(\mathbb{R}))$ is a bijection satisfying conditions in Definition 3. Thus $\mathbb{P} \sqcup \mathbb{R} \equiv_m \mathbb{Q} \sqcup \mathbb{R}$. \square

It is instructive to consider a potentially stronger variant of \equiv_m defined in analogy to \equiv_s [16]: $\mathbb{P} \equiv_m^s \mathbb{Q}$ iff $\mathbb{P} \sqcup \mathbb{R} \equiv_m \mathbb{Q} \sqcup \mathbb{R}$ holds for all \mathbb{R} such that $\mathbb{P} \sqcup \mathbb{R}$ and $\mathbb{Q} \sqcup \mathbb{R}$ are defined. However, Theorem 2 implies that \equiv_m^s adds nothing to \equiv_m since $\mathbb{P} \equiv_m^s \mathbb{Q}$ iff $\mathbb{P} \equiv_m \mathbb{Q}$.

As regards the computational complexity, deciding \equiv_m is **coNP**-hard in general, since deciding $P \equiv Q$ reduces to deciding $(P, \emptyset, \text{Hb}(P)) \equiv_m (Q, \emptyset, \text{Hb}(Q))$. If $\text{Hb}_h(P) = \text{Hb}_h(Q) = \emptyset$, deciding $\mathbb{P} \equiv_m \mathbb{Q}$ reduces to deciding $P \equiv_u^I Q$ [25]. Thus deciding \equiv_m is **coNP**-complete in the fully visible case. In the other extreme, if $\text{Hb}_v(\mathbb{P}) = \text{Hb}_v(\mathbb{Q}) = \emptyset$, then $\mathbb{P} \equiv_m \mathbb{Q}$ iff $|\text{SM}(\mathbb{P})| = |\text{SM}(\mathbb{Q})|$. This suggests a much higher computational complexity of deciding \equiv_m in general because classical models can be captured with stable models [20] and the counting problem #SAT is **#P**-complete [24].

A way to govern the computational complexity is to limit the use of hidden atoms as in the case of \equiv_v [15]. Therefrom we adopt the property of having *enough visible atoms* (EVA). For an interpretation $M_v \subseteq \text{Hb}_v(P)$ for the visible part of a NLP P , the *hidden part* P_h/M_v of P relative M_v contains $h \leftarrow B_h^+, \sim B_h^-$ for each rule $h \leftarrow B^+, \sim B^- \in P$ such that $h \in \text{Hb}_h(P)$ and $M_v \models B_h^+ \cup \sim B_h^-$.

Definition 11 *A NLP P has enough visible atoms iff P_h/M_v has a unique stable model for every interpretation $M_v \subseteq \text{Hb}_v(P)$.*

The idea behind the EVA property is that the interpretation of $\text{Hb}_h(P)$ is uniquely determined for each interpretation of $\text{Hb}_v(P)$. Consequently, the stable models of P can be distinguished from each other on the basis of their visible parts. By the EVA assumption [15], the verification of \equiv_v becomes **coNP**-complete for SMODELS programs² involving hidden atoms—enabling the translation-based method [14] for \equiv_v . Although verifying the EVA property can be hard in general, there are syntactic subclasses of NLPs (e.g. those for which P_h/M_v is always stratified) with the EVA property. It should be stressed that the use of visible atoms is not limited in this setting.

² This class of programs includes normal logic programs.

As for \equiv_v , the EVA assumption is equally important in conjunction with \equiv_m which becomes evident once we work out the interconnections of \equiv_v and \equiv_m . We begin by describing ways in which modular equivalence can be exploited in the verification of visible/weak equivalence. One concrete step in this respect is to reduce the problem of verifying \equiv_m to that of \equiv_v by introducing a special module \mathbb{G}_I that acts as a context generator in analogy to [25].

Theorem 3 *If $\text{Hb}_v(\mathbb{P}) = \text{Hb}_v(\mathbb{Q}) = O \cup I$ holds for \mathbb{P} and \mathbb{Q} , then $\mathbb{P} \equiv_m \mathbb{Q}$ iff $\mathbb{P} \sqcup \mathbb{G}_I \equiv_v \mathbb{Q} \sqcup \mathbb{G}_I$ where $\mathbb{G}_I = (\{a \leftarrow \sim \bar{a}, \bar{a} \leftarrow \sim a \mid a \in I\}, \emptyset, I)$ generates all possible inputs for \mathbb{P} and \mathbb{Q} .*

Proof sketch. Note that \mathbb{G}_I has $2^{|I|}$ stable models of the form $A \cup \{\bar{a} \mid a \in I \setminus A\}$ where $A \subseteq I$. Thus $\mathbb{P} \equiv_v \mathbb{P} \sqcup \mathbb{G}_I$ and $\mathbb{Q} \equiv_v \mathbb{Q} \sqcup \mathbb{G}_I$ follow by Definitions 2 and 3 and Theorem 1. It follows that $\mathbb{P} \equiv_m \mathbb{Q}$ iff $\mathbb{P}(A) \equiv_v \mathbb{Q}(A)$ for all $A \subseteq I$ iff $\mathbb{P} \sqcup \mathbb{G}_I \equiv_v \mathbb{Q} \sqcup \mathbb{G}_I$. \square

Consequently, the translation-based method from [15] can be used to decide $\mathbb{P} \equiv_m \mathbb{Q}$ given that \mathbb{P} and \mathbb{Q} have enough visible atoms (\mathbb{G}_I has the EVA property trivially). The task is to show that $\text{EQT}(\mathbb{P} \sqcup \mathbb{G}_I, \mathbb{Q} \sqcup \mathbb{G}_I)$ and $\text{EQT}(\mathbb{Q} \sqcup \mathbb{G}_I, \mathbb{P} \sqcup \mathbb{G}_I)$ have no stable models.

The introduction of \equiv_m was much motivated by the need of modularizing the verification of \equiv .³ To make this idea clear, we propose a strategy to utilize \equiv_m in the task of verifying the visible/weak equivalence of P and Q . It is required that the module structure for P and Q is either specified explicitly or detected automatically by computing the strongly connected components of $\text{Dep}^+(P)$ and $\text{Dep}^+(Q)$. Assuming that Q is obtained from P through local modifications, it is likely that these components are pairwise compatible and we can partition P and Q so that $P = \mathbb{P}_1 \sqcup \dots \sqcup \mathbb{P}_n$ and $Q = \mathbb{Q}_1 \sqcup \dots \sqcup \mathbb{Q}_n$ where \mathbb{P}_i and \mathbb{Q}_i have the same input and output sets for all i and $\mathbb{P}_i = \mathbb{Q}_i$ might hold for a number of i 's. Now, verifying $\mathbb{P}_i \equiv_m \mathbb{Q}_i$ for every i is not of interest as $\mathbb{P}_i \not\equiv_m \mathbb{Q}_i$ does not necessarily imply $P \not\equiv_v Q$. However, the verification of $P \equiv_v Q$ can still be organized as a sequence of n module-level tests

$$\left(\bigsqcup_{j=1}^{i-1} \mathbb{Q}_j \right) \sqcup \mathbb{P}_i \sqcup \left(\bigsqcup_{j=i+1}^n \mathbb{P}_j \right) \equiv_m \left(\bigsqcup_{j=1}^{i-1} \mathbb{Q}_j \right) \sqcup \mathbb{Q}_i \sqcup \left(\bigsqcup_{j=i+1}^n \mathbb{P}_j \right) \quad (1)$$

where $1 \leq i \leq n$. The resulting chain of equalities conveys $P \equiv_v Q$ under the assumption that P and Q have a completely specified input. If not, then \equiv_m can be addressed in a similar fashion using (1).

Example 4 Consider $P = \mathbb{P}_1 \sqcup \mathbb{P}_2$ and $Q = \mathbb{Q}_1 \sqcup \mathbb{Q}_2$ where $\mathbb{P}_1 = (\{c \leftarrow \sim a.\}, \{a, b\}, \{c\})$, $\mathbb{P}_2 = (\{a \leftarrow b.\}, \emptyset, \{a, b\})$, $\mathbb{Q}_1 = (\{c \leftarrow \sim b.\}, \{a, b\}, \{c\})$ and $\mathbb{Q}_2 = (\{b \leftarrow a.\}, \emptyset, \{a, b\})$. Now $\mathbb{P}_1 \not\equiv_m \mathbb{Q}_1$ but $\mathbb{P}_1 \equiv_v \mathbb{Q}_1$ for all inputs generated by \mathbb{P}_2 and \mathbb{Q}_2 . Thus $\mathbb{P}_1 \sqcup \mathbb{P}_2 \equiv_m \mathbb{Q}_1 \sqcup \mathbb{Q}_2 \equiv_m \mathbb{Q}_1 \sqcup \mathbb{Q}_2$, $P \equiv_v Q$ and $P \equiv Q$.

The programs involved in each test (1) differ in \mathbb{P}_i and \mathbb{Q}_i for which the other modules form a common context, say \mathbb{C}_i . A way to optimize the verification of $\mathbb{P}_i \sqcup \mathbb{C}_i \equiv_m \mathbb{Q}_i \sqcup \mathbb{C}_i$ is to view \mathbb{C}_i as a module generating input for \mathbb{P}_i and \mathbb{Q}_i and to adjust the method from [15] to use $\text{EQT}(\mathbb{P}_i, \mathbb{Q}_i) \sqcup \mathbb{C}_i$ rather than $\text{EQT}(\mathbb{P}_i \sqcup \mathbb{C}_i, \mathbb{Q}_i \sqcup \mathbb{C}_i)$. We expect computational advantage from this strategy especially when the context \mathbb{C}_i is clearly larger than the modules \mathbb{P}_i and \mathbb{Q}_i .

6 RELATED WORK

The notion of modular equivalence is already contrasted with other equivalence relations in Sections 3 and 5.

³ Recall that \equiv_v coincides with \equiv for programs P and Q having equal and fully visible Herbrand bases.

Bugliesi et al. [2] present an extensive survey of modularity in conventional logic programming. Two mainstream programming disciplines can be identified: *programming-in-the-large* where programs are composed with algebraic operators and *programming-in-the-small* with abstraction mechanisms. Our approach belongs to the former discipline due to resemblance to [10] but stable model semantics and the denial of positive recursion between modules can be pointed out as obvious differences. Eiter et al. [6] present a programming-in-the-small approach to ASP and view program modules as *generalized quantifiers* the definitions of which are allowed to nest, i.e. P can refer to another module Q by using it as a generalized quantifier.

A variety of module conditions (cf. Definition 7) have been introduced. Maher [18] forbids all recursion between modules and considers Przymusinski's *perfect models* rather than stable models. Brogi et al. [1] employ operators for program composition and visibility conditions that correspond to the second item in Definition 7. However, their approach covers only positive programs and the least model semantics. Etalle and Gabbrielli [8] also seek congruence relations but for *constraint logic programs*. Their module architecture is based on a condition similar to ours, i.e. $\text{Hb}(P) \cap \text{Hb}(Q) \subseteq \text{Hb}_v(P) \cap \text{Hb}_v(Q)$, but no distinction of input and output is made.

Eiter, Gottlob, and Mannila [5] consider *disjunctive logic programs* as a query language for relational databases. Thus normal programs with variables are covered as a special case. A query program π is instantiated with respect to an input database D confined by an input schema \mathbf{R} . The semantics of π determines e.g. the stable models of $\pi[D]$ which are projected with respect to an output schema \mathbf{S} . In view of our work, π can be understood as a program module \mathbb{P} with input I and output O based on \mathbf{R} and \mathbf{S} , respectively, so that $\pi[D]$ corresponds to $\mathbb{P}(D)$. Their module architecture enables a generalization of splitting sets as both *positive and negative dependencies* are taken into account and no inter-module recursion is tolerated.

Faber et al. [9] apply the *magic set method* in the evaluation of Datalog programs with negation, i.e. effectively normal programs. This involves the concept of an *independent set* S of a program P which is a specialization of a splitting set. The rough idea is that S gives rise to a *module* $T = \{h \leftarrow B^+, \sim B^- \in P \mid h \in S\}$ of P so that $T \subseteq P$ and $\text{Head}(T) = S$ and the semantics of S is not affected by $P \setminus T$. Like splitting sets, independent sets are not that flexible in parceling NLPs. E.g., the splitting achieved in Example 2 is impossible. In view of our results, the account of *dangerous rules* unnecessarily pushes negative recursion inside modules. Moreover, the module theorem of Faber et al. [9] is weaker than Theorem 1.

7 CONCLUSION

In this article, we propose a module architecture for answer set programming. The compatibility of the module system and stable models is achieved by denying positive recursion between modules. A number of interesting results is obtained. First, the splitting set theorem [17] is generalized to a case where negative recursion is allowed between modules. Second, the respective equivalence relation \equiv_m is a proper congruence relation for the join operation \sqcup between modules. Third, the verification of modular equivalence can be accomplished with existing methods so that specialized solvers need not be developed. Last but not least, we have a preliminary plan how the task of verifying weak equivalence can be modularized using \equiv_m .

Yet the potential gain from the modular verification strategy has to be evaluated by conducting experiments. A further theoretical question is how the existing model theory based on *SE-models* and *UE-models* [3] is tailored to the case of modular equivalence. There is

also a need to expand the module architecture and module theorem proposed here to cover other classes of logic programs such as e.g. weight constraint programs, and disjunctive/nested programs.

ACKNOWLEDGEMENTS

This research has been partially funded by the Academy of Finland (project #211025). The first author acknowledges the financial support from Helsinki Graduate School in Computer Science and Engineering, Nokia Foundation, and Finnish Cultural Foundation.

REFERENCES

- [1] A. Brogi, P. Mancarella, D. Pedreschi, and F. Turini, 'Modular logic programming', *ACM Trans. Program. Lang. Syst.*, **16**(4), 1361–1398, (1994).
- [2] M. Bugliesi, E. Lamma, and P. Mello, 'Modularity in logic programming', *J. Log. Program.*, **19/20**, 443–502, (1994).
- [3] T. Eiter and M. Fink, 'Uniform equivalence of logic programs under the stable model semantics', in *Proc. ICLP'03*, pp. 224–238, (2003).
- [4] T. Eiter, M. Fink, H. Tompits, and S. Woltran, 'Simplifying logic programs under uniform and strong equivalence', in *Proc. LPNMR'04*, pp. 87–99, (2004). LNAI 2923.
- [5] T. Eiter, G. Gottlob, and H. Mannila, 'Disjunctive datalog.', *ACM Trans. Database Syst.*, **22**(3), 364–418, (1997).
- [6] T. Eiter, G. Gottlob, and H. Veith, 'Modular logic programming and generalized quantifiers', in *Proc. LPNMR'97*, pp. 290–309, (1997).
- [7] T. Eiter, H. Tompits, and S. Woltran, 'On solution correspondences in answer-set programming', in *Proc. IJCAI'05*, pp. 97–102, (2005).
- [8] S. Etalle and M. Gabbrielli, 'Transformations of CLP modules', *Theor. Comput. Sci.*, **166**(1–2), 101–146, (1996).
- [9] W. Faber, G. Greco, and N. Leone, 'Magic sets and their application to data integration.', in *Proc. ICDT'05*, pp. 306–320, (2005). LNCS 3363.
- [10] H. Gaifman and E. Shapiro, 'Fully abstract compositional semantics for logic programs', in *Proc. POPL'89*, pp. 134–142, (1989).
- [11] M. Gelfond and N. Leone, 'Logic programming and knowledge representation — the A-Prolog perspective', *Artif. Intell.*, **138**, 3–38, (2002).
- [12] M. Gelfond and V. Lifschitz, 'The stable model semantics for logic programming', in *Proc. ICLP'88*, pp. 1070–1080, (1988).
- [13] T. Janhunen, 'Translatability and intranslatability results for certain classes of logic programs', Research report A82, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, (2003).
- [14] T. Janhunen and E. Oikarinen, 'Testing the equivalence of logic programs under stable model semantics', in *Proc. JELIA'02*, pp. 493–504, (2002). LNAI 2424.
- [15] T. Janhunen and E. Oikarinen, 'Automated verification of weak equivalence within the SMODELS system', (2005). Submitted to *TPLP*.
- [16] V. Lifschitz, D. Pearce, and A. Valverde, 'Strongly equivalent logic programs', *ACM Trans. Comput. Log.*, **2**(4), 526–541, (2001).
- [17] V. Lifschitz and H. Turner, 'Splitting a logic program', in *Proc. ICLP'94*, pp. 23–37, (1994).
- [18] M. J. Maher, 'A transformation system for deductive database modules with perfect model semantics', *Theor. Comput. Sci.*, **110**(2), 377–403, (1993).
- [19] V. W. Marek and M. Truszczyński, 'Stable models and an alternative logic programming paradigm', in *The Logic Programming Paradigm: a 25-Year Perspective*, 375–398, Springer, (1999).
- [20] I. Niemelä, 'Logic programming with stable model semantics as a constraint programming paradigm', *Ann. Math. Artif. Intell.*, **25**(3–4), 241–273, (1999).
- [21] E. Oikarinen and T. Janhunen, 'Verifying the equivalence of logic programs in the disjunctive case', in *Proc. LPNMR'04*, pp. 180–193, (2004). LNAI 2923.
- [22] Y. Sagiv, 'Optimizing datalog programs', in *Proc. PODS'87*, pp. 349–362, (1987).
- [23] H. Turner, 'Strong equivalence made easy: nested expressions and weight constraints.', *TPLP*, **3**(4–5), 609–622, (2003).
- [24] L. G. Valiant, 'The complexity of enumeration and reliability problems', *SIAM J. Comput.*, **8**(3), 410–421, (1979).
- [25] S. Woltran, 'Characterizations for relativized notions of equivalence in answer set programming', in *Proc. JELIA'04*, pp. 161–173, (2004).

Preference representation with 3-points intervals

Öztürk Meltem and Tsoukiàs Alexis¹

Abstract. In this article we are interested in the representation of qualitative preferences with the help of 3-points intervals (a vector of three increasingly ordered points). Preferences are crucial when an agent has to autonomously make a choice over several possible actions. We provide first of all an axiomatization in order to characterize our representation and then we construct a general framework for the comparison of 3-points intervals. Our study shows that from the fifteen possible different ways to compare 3-points intervals, seven different preference structures can be defined, allowing the representation of sophisticated preferences. We show the usefulness of our results in two classical problematics: the comparison of alternatives and the numerical representation of preference structures. Concerning the former one, we propose procedures to construct non classical preference relations (intransitive preferences for example) over objects being described by three ordered points. Concerning the latter one, assuming that preferences on the pairwise comparisons of objects are known, we show how to associate a 3-points interval to every object, and how to define some comparison rules on these intervals in order to have a compact representation of preferences described with these pairwise comparisons.

1 Introduction

The notion of preference, initially introduced by economists ([3, 6]) and researchers on Decision Making (DM) ([13, 11, 18, 17, 15]), has recently received an increasing attention in AI where artificial agents play the role of automated decision makers ([19, 7]).

In DM, preferences are used for two different problematics ([21]): the *comparison problem* and the *numerical representation problem*. These two problems arise naturally in AI since comparing objects and establishing preference (or any other order relations) is a key issue in knowledge representation and elicitation.

The *comparison problem* deals with the construction of preference relations over each pair of alternatives. In such a case evaluations of alternatives are known and may have different nature: numbers, colors, symbols, figures, intervals, fuzzy numbers, etc. The construction of relations may not be an easy task even with quantitative evaluations. For instance, consider a maximization problem with three alternatives (a , b and c) evaluated by numbers ($g(a) = 25$, $g(b) = 11$ and $g(c) = 9$). Depending on the context and/or the decision maker, we may have different relations. One solution may be to say that there are only strict preferences (a is strictly preferred to b and c (aPb and aPc) and b is strictly preferred to c (bPc)) while in a different context the alternatives b and c may be considered as indifferent (bIc) since the difference between their evaluation is not significant. It is clear that the relations obtained in the two different contexts do not have the same properties and they do not lead to the same model.

The numerical representation problem goes in the opposite way. The preference on each pair of alternatives being known, the problem is to check if there exists (and under which conditions) one or more real valued functions which, when applied to the set of alternatives, will return the preferences of the decision maker. As an example, consider three alternatives a , b and c for which the decision maker claims that he is indifferent between c and b and he strictly prefers c to a and b to a . There are several different numerical representations which could account for such preferences. One option may be to associate intervals $[0, 1]$ to a , $[2, 4]$ to b and $[3, 6]$ to c under the rules “ x is preferred to y iff the interval of x is completely to the right of the interval of y (no intersection) ”.

We consider both types of problems with a special attention to an interval representation. Comparing intervals is a problem relevant to several disciplines. We need intervals in order to take into account intransitivity of indifference due to the presence of one or more thresholds, to compare time intervals ([2]), or to represent imprecision or uncertainty (the price of x lies between A and B, the quality of y lies between “medium” and “good” ...). In this article, we make use of a special type of intervals that we call “3-points intervals” (intervals with an intermediate point). Such intervals contain only ordinal information (the distances between points are not important) which allows us to represent qualitative evaluations. Qualitative approaches become more and more attractive in AI since the only existing knowledge may be qualitative or it may be easier to get qualitative information from experts or qualitative rules may be easier and faster (see [4] and [5]).

The main contribution of this article is to propose a general framework for the comparison of 3-points intervals. The general advantage of these intervals is their capacity of representation, especially for sophisticated preferences. Our results are useful for both of the problematics. Concerning the comparison problem, our work shows how to compare two intervals having only ordinal information in order to fit some desired properties such as transitivity of preference, intransitivity of indifference etc. Concerning the numerical representation problem, there are two main advantages. First of all the use of 3-points intervals allows to represent complex preferences. For instance, the use of simple numbers remains inefficient in the majority of cases (only total orders and weak orders have a representation with numbers), such a reason has led to the use of intervals for different preference structures ([12, 8, 20, 16]). There are many results concerning the classical intervals (2-points intervals), however such intervals may appear insufficient face to more complex preferences (for example when the preference is intransitive). For that reason we are interested in 3-points intervals for which there is a limited number of research ([10]). Another advantage is related to the cardinality of the set of alternatives. When there are too many alternatives (let n be the number of alternatives), it can be preferable to stock only the 3-points interval representation of each alternative ($3 * n$ informa-

¹ LAMSADE - CNRS, Université Paris Dauphine, email: {ozturm, tsoukiás}@lamsade.dauphine.fr

tion) instead of stocking all the pairwise comparisons of alternatives ($\frac{n(n-1)}{2}$ information). From this point of view we can say that 3-points interval representation proposes a compact representation for complex preferences.

We organize the paper in the following way: in section 2 we introduce basic notations, we propose an axiomatization for the characterization of the 3-points interval representation. A general type of representation satisfying such axioms are also presented in this section. In section 3 we propose an exhaustive analysis of all the preference structures having a 3-points interval representation and in section 4 we conclude with some future research directions.

2 Basic notions and 3-points interval representation

In this paper we study complete preference structures with two binary relations: the strict preference relation P which is an asymmetric relation and the indifference relation I which is the symmetric complement of P . We introduce first of all some notions that we will use in the axiomatization.

We call a “3-points interval” an interval $x = [f_1(x), f_3(x)]$ with an intermediate point $f_2(x)$ (i.e. $f_1(x) < f_2(x) < f_3(x)$).

Then, we introduce a new notion that we call the “relative position” and that we denote by φ . The notation $\varphi(x, y)$ represents the position of the interval x with respect to the interval y ($\varphi(x, y) \neq \varphi(y, x)$).

Definition 1 (Relative position) *The relative position $\varphi(x, y)$ is the 3-tuple $(\varphi_1(x, y), \varphi_2(x, y), \varphi_3(x, y))$ where $\varphi_i(x, y)$ represents the number of j such that $f_i(x) < f_j(y)$.*

Intuitively, φ represents to what extend the position of two intervals is close to the case of two disjoint intervals, case which guarantees a strict preference. The following example illustrates the previous definition.

Example 1 *Let x and y be two 3-points intervals represented in figure 1, then $\varphi(x, y) = (1, 0, 0)$. $\varphi_1(x, y) = 1$ since there is only $f_3(y)$ being greater than $f_1(x)$ and $\varphi_2(x, y) = \varphi_3(x, y) = 0$ since $f_2(x)$ and $f_3(x)$ are greater than all the points of y .*

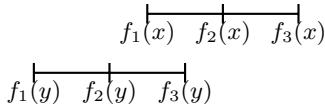


Figure 1. Relative position $\varphi(x, y) = (1, 0, 0)$

Let us remark that there are $20 \binom{(2*3)!}{(3!)^2}$ different relative positions when two 3-points intervals are compared.

The strict preference between two intervals depends on their relative positions and naturally there are some relative positions which are more suitable for the representation of a strict preference than others. For example the case where two intervals are disjoint is more suitable for a strict preference than a case where one interval is included to another. For such a purpose we introduce a new binary relation, called “stronger than”, on the set of relative positions.

Definition 2 (“Stronger than” relation) *Let φ and φ' be two relative positions, then we say that φ is “stronger than” φ' and note $\varphi \triangleright \varphi'$ if $\forall i \in \{1, \dots, n\}, \varphi_i \leq \varphi'_i$.*

We present an example showing how we define a “stronger than” relation.

Example 2 *Let $\varphi(x, y)$ and $\varphi(x, t)$ be two relative positions of the figure 2. We have $\varphi(x, y) = (1, 1, 0)$, $\varphi(x, t) = (2, 1, 0)$. We get “ $\varphi(x, y)$ is stronger than $\varphi(x, t)$ ” since $1 \leq 2$, $1 \leq 1$ and $0 \leq 0$.*

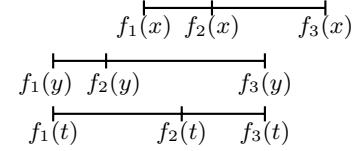


Figure 2. Example: $(1, 1, 0) \triangleright (2, 1, 0)$

The “stronger than” relation satisfies some classical properties:

Proposition 1 \triangleright is a partial order (reflexive, antisymmetric and transitive) defining a lattice on the set of possible relative positions.

Proof. \triangleright is a partial order since it is induced from the relation “ $<$ ” which is reflexive, antisymmetric and transitive. ■

Let us remark that the relation \triangleright is not complete: for example we have $(2, 0, 0) \not\triangleright (1, 1, 0)$ and $(1, 1, 0) \not\triangleright (2, 0, 0)$. We present in figure 3 the graph of the relation \triangleright .

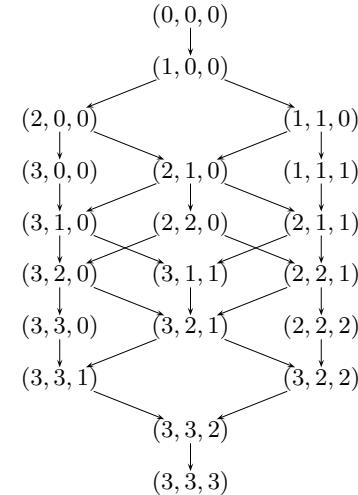


Figure 3. Graph of the stronger than relation

We are ready now to define the strict preference relation P and the indifference relation I . We will define P as a set of relative positions, satisfying some constraints, and construct I as the complement of P . For such a purpose we propose an axiomatization:

Axiome 1 *The relation $P \cup I$ is complete and I is the complement of P (i.e. $I(x, y) \Leftrightarrow \neg P(x, y) \wedge \neg P(y, x)$).*

Axiome 2 *The relations $P(x, y)$ and $I(x, y)$ depends only on the relative position of x and y .*

Axiome 3 *If a relative position φ is in the set of the strict preference P then all the relative positions which are stronger than φ are also in the set of P .*

Axiome 4 *If for all i , $f_i(x) < f_i(y)$ then $P(x, y)$ is not satisfied.*

Axiome 5 The set of relative positions forming P has one and only one weakest relative position (relative position which is weak than every relative position of the set).

Axiom 1 shows that P and I are exhaustive and exclusive, axiom 2 presents the comparison parameters and axiom 3 guarantees the monotonicity. Every relative position is not a good candidate to represent a strict preference. Axiom 4 eliminates some undesired situations in the definition of P . The role of the strongest relative position of a set of P is very important since we can determine all the other elements of the set by the help of the strongest one. Axiom 5 guarantees a unique representation for the strict preference relations by forbidding the existence of more than one strongest relative positions in their set.

It is easy to calculate the number of sets satisfying such axioms. Since every set has just one strongest relative position, every relative position may present one set, of course except the ones which do not satisfy the axiom 4. The number of relative positions with “for all i , $f_i(x) < f_i(y)$ ” is five ($\frac{1}{3+1} \binom{6}{3}$). Since there are, in total, twenty relative positions, the number of sets satisfying axioms 1-5 is fifteen.

We can present now the 3-points interval representation of a preference structure satisfying axioms 1-5. First of all, let's give a formal definition of the preference structure induced by the different possible relative positions of 3-points intervals.

Definition 3 Let $\varphi = (\varphi_1, \varphi_2, \varphi_3)$ be a 3-tuple in $\{0, 1, 2, 3\}$, and x and y two 3-points intervals. The preference relations $P_{\leq\varphi}, I_{\leq\varphi}$ associated to φ is defined as

$$\begin{aligned} P_{\leq\varphi}(x, y) &\iff \varphi(x, y) \triangleright \varphi \\ I_{\leq\varphi}(x, y) &\iff \neg P_{\leq\varphi}(x, y) \wedge \neg P_{\leq\varphi}(y, x) \end{aligned}$$

Now, consider the preference relation $P_{\leq(2,0,0)}$. Then $P_{\leq(2,0,0)}(x, y)$ iff $f_1(y) < f_1(x)$, $f_3(y) < f_2(x)$ and $f_3(y) < f_3(x)$. We can remark that the third inequality is redundant. This motivates the definition of the component set of a triple φ .

Definition 4 Let $\varphi = (\varphi_1, \varphi_2, \varphi_3)$ be a 3-tuple in $\{0, 1, 2, 3\}$. The component set $Cp_{\leq\varphi}$ associated to φ is the set of couples $(3 - \varphi_i, i)$ such that there is no $i' < i$ with $\varphi_{i'} \leq \varphi_i$.

For instance, $Cp_{\leq(2,0,0)} = \{(1, 1), (3, 2)\}$. Hence, $Cp_{\leq\varphi}$ represents the set of couples of points that are sufficient to be compared. Conditions on the elements of $Cp_{\leq\varphi}$ guarantees the minimality of the representation. The set $Cp_{\leq\varphi}$ contains all the information concerning the preference structure.

It is easy to verify that the preference structure associated to a triple φ verifies axioms 1,2,3 and 5. Following definition 4, one can show that axiom 4 is verified by $P_{\leq\varphi}$ iff $Cp_{\leq\varphi}$ contains at least one (i, j) with $i \geq j$.

3 3-points interval comparisons

In this section we analyze in details the fifteen sets of P satisfying our axiomatization. Let us remind that each set represents a strict preference relation which has a 3-points interval representation and the component set $Cp_{\leq\varphi}$ has the whole information about this representation.

Our study shows that from the fifteen sets of P , seven different preference structures can be defined, some of them having more than

one 3-points interval representation. For the sake of clarity, we will first of all present the classical definition of these seven structures, then give their equivalent characterization with 3-points intervals by the help of component sets and finally present in table 1 all the 3-points interval representations of these preference structures.

An exhaustive study of the 3-points intervals shows that weak orders and bi-weak orders have three different 3-points interval representations while three-weak orders have one, interval orders have three, split interval orders have one, triangle orders have two and intransitive orders have two. We present first of all the definition of each preference structure that we cited:

Let P be a binary relation on a finite set A and I be the symmetric complement of P , then

- $P \cup I$ is a *weak order* if and only if there exists a real-valued function f defined on A such that

$$\forall x, y \in A, xPy \iff f(x) > f(y)$$
- $P \cup I$ is a *bi-weak order* if and only if there exist two different real-valued functions f_1 and f_2 defined on A such that

$$\forall x, y \in A, xPy \iff \begin{cases} f_1(x) > f_1(y) \\ f_2(x) > f_2(y) \end{cases}$$

It is easy to see that bi-weak orders are defined as the intersection of two weak orders.
- $P \cup I$ is a *3-weak order* if and only if it is defined as the intersection of three weak orders.
- $P \cup I$ is an *interval order* if and only if there exist two real-valued functions f_1 and f_2 , defined on A such that

$$\begin{cases} \forall x, y \in A, xPy \iff f_1(x) > f_2(y) \\ \forall x \in A, f_2(x) > f_1(x) \end{cases}$$
- $P \cup I$ is a *split interval order* if and only if there exist three real-valued functions f_1 , f_2 and f_3 defined on A such that

$$\begin{cases} \forall x, y \in A, xPy \iff \begin{cases} f_1(x) > f_2(y), \\ f_2(x) > f_3(y), \\ \forall x \in A, f_3(x) > f_2(x) > f_1(x) \end{cases} \end{cases}$$
- $P \cup I$ is a *triangle order* if and only if it is defined as the intersection of one weak order and one interval order.
- $P \cup I$ is an *intransitive order* if and only if P is intransitive.

From these seven structures weak orders are the most used ones. Their difference from linear orders (total orders) comes from the fact that weak orders may have equivalence classes (two different objects may be considered as indifferent) which is forbidden in the case of linear orders. Bi-weak orders are also known structures, especially for the researchers of DM. They are equivalent to bilinear orders (interested reader may find more details in [9]). Three-weak orders were born from the generalization of bi-weak orders (for more details see [14]). Interval orders have been introduced by Fishburn ([8]). The relaxation of the coherence condition of semiorders (semiorders have an interval representation where each interval has the same length) has led to interval orders which are especially used in the presence of discrimination thresholds in order to represent intransitive indifference. Split interval orders are especially studied by mathematicians ([10]) and allow the representation of very sophisticated preferences. The name of triangle orders comes from their classical representa-

tion: an object is preferred to another one if and only if the triangle representing the first object is completely to the right of the triangle representing the second one (no intersection)(more details can be found in [14]). Intransitive orders are marginal orders, however they are used in some special domains (such as the biology in the case of cellule comparison or the chemistry in the case of molecular connection [1]). Circles are used in order to represent such structures: an object is preferred to another one if and only if the circle representing the first object is completely to the right of the circle representing the second one (circles may have different diameters). Unfortunately, we can not give here more details about these seven structures, interested reader may find more information in the cited references.

Let us remark that the classical representation of the majority of these structures do not make use of intervals (intervals can be seen as vectors of some ordered points). For instance weak orders use simple numbers while bi-weak orders (resp. three-weak orders) utilize two, not necessarily ordered numbers (resp. three points) (for instance we can have $f_1 < f_2$ or $f_2 < f_1$). Triangle orders are represented by triangles and intransitive orders by circles. Our study shows that all these seven structures have a 3-points interval representation. We will present now the general form of these representations by the help of component sets. We begin by some propositions concerning the transitivity properties since they are fundamental for some preference structures :

- $P_{\leq\varphi}$ is transitive if and only if $\forall(i, j) \in Cp_{\leq\varphi}, i \geq j$,
- $I_{\leq\varphi}$ is transitive if and only if $\exists i, Cp_{\leq\varphi} = \{(i, i)\}$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a weak order if and only if $\exists i, Cp_{\leq\varphi} = \{(i, i)\}$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a bi-weak order if and only if $|Cp_{\leq\varphi}| = 2$ and $\forall(i, j) \in Cp_{\leq\varphi}, i = j$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a 3-weak order if and only if $|Cp_{\leq\varphi}| = 3$ and $\forall(i, j) \in Cp_{\leq\varphi}, i = j$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is an interval order if and only if $Cp = \{(i, j)\}$ where $i \geq j$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a triangle order if and only if $Cp_{\leq\varphi} = \{(l, l), (i, j)\}$, where $i \geq j$,
- $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a intransitive order if and only if $\exists(i, j) \in Cp_{\leq\varphi}, i < j$.

We present now the key-steps of the proofs of these general propositions.

- **The transitivity of $P_{\leq\varphi}$:**
 - If $\forall(i, j) \in Cp_{\leq\varphi}, i \geq j$ then $P_{\leq\varphi}$ is transitive: obvious.
 - If $P_{\leq\varphi}$ is transitive then $\forall(i, j) \in Cp_{\leq\varphi}, i \geq j$: we prove this result by showing that if $\exists(i, j) \in Cp_{\leq\varphi} i < j \implies \exists x, y, z, P_{\leq\varphi}(x, y) \wedge P_{\leq\varphi}(y, z)$ and $\neg P_{\leq\varphi}(x, z)$.
- **The transitivity of $I_{\leq\varphi}$:**
 - $Cp_{\leq\varphi} = \{(i, i)\}$ implies $I_{\leq\varphi}$ is transitive: obvious.
 - $I_{\leq\varphi}$ is transitive implies $Cp_{\leq\varphi} = \{(i, i)\}$: we prove this result by contradiction. Supposing that $I_{\leq\varphi}$ is transitive we analyze two different cases: $\exists(i, j) \in Cp_{\leq\varphi}, i \neq j$, and $\forall(i, j) \in Cp_{\leq\varphi}, i = j$ and $|Cp_{\leq\varphi}| > 1$. We show that these two

cases are contradictory with the transitivity of $I_{\leq\varphi}$.

- **Weak order:**

- If $Cp_{\leq\varphi} = \{(i, i)\}$ then $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a weak order: we prove that $I_{\leq\varphi}$ and $P_{\leq\varphi}$ are transitive and $P_{\leq\varphi} \cup I_{\leq\varphi}$ is reflexive and complete.

- If $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a weak order then $Cp_{\leq\varphi} = \{(i, i)\}$: the key idea is the transitivity of $I_{\leq\varphi}$. If $P_{\leq\varphi} \cup I_{\leq\varphi}$ is a weak order then $I_{\leq\varphi}$ is transitive and if $I_{\leq\varphi}$ is transitive then $Cp_{\leq\varphi} = \{(i, i)\}$.

- **Bi-weak order:** the proof follows directly from the one of weak orders.

- **Three-weak order:** the proof follows directly from the one of weak orders.

- **Interval order:** for this proof we make use of the relational characterization of an interval order: $P \cup I$ is an interval order if

$$\begin{cases} P.I.P \subset P, \\ P \cup I \text{ is reflexive and complete.} \end{cases}$$

where $P.I.P \subset P$ means $\forall x, y, z, t$, if $P(x, y) \wedge I(y, z) \wedge P(z, t)$ then $P(x, t)$.

We prove first of all that $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$ iff $Cp = \{(i, j)\}$ where $i \geq j$:

- If $Cp = \{(i, j)\}$ where $i \geq j$ then $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$: obvious.

- If $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$ then $Cp = \{(i, j)\}$ where $i \geq j$: first of all if $Cp = \{(i, j)\}$ with $i < j$ then $P_{\leq\varphi}$ is not transitive. In this case it is easy to see that when $I_{\leq\varphi}$ is the identity $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$ is not satisfied. We prove then that if $|Cp_{\leq\varphi}| > 1$ then $\neg(P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi})$. We analyze two cases where $|Cp_{\leq\varphi}| > 1$: $\exists(i, j) \in Cp_{\leq\varphi}, i < j$ and $\forall(i, j) \in Cp_{\leq\varphi}, i \geq j$. The first one provides an intransitive $P_{\leq\varphi}$. The key point of the analysis of the second case is the definition of $I_{\leq\varphi}$ when $|Cp_{\leq\varphi}| > 1$: let $(i, j), (l, m)$ be elements of $Cp_{\leq\varphi}$ then $f_i(x) \geq f_j(y) \wedge f_l(y) \geq f_m(x)$ with $(i, j) \neq (l, m) \implies I_{\leq\varphi}(x, y)$. It is easy to see that this implication has one part where a point of x is greater than a point of y and another part which inverses such inequality. In this case one can always find four elements w, x, y, z such that $P_{\leq\varphi}(w, x), I_{\leq\varphi}(x, y), P_{\leq\varphi}(y, z)$ and $\neg P_{\leq\varphi}(w, z)$.

We can now proof the characterization of interval orders:

- If $Cp = \{(i, j)\}$ where $i \geq j$ then $P_{\leq\varphi} \cup I_{\leq\varphi}$ is an interval order: we prove that $P_{\leq\varphi} \cup I_{\leq\varphi}$ is reflexive and complete and $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$.

- If $P_{\leq\varphi} \cup I_{\leq\varphi}$ is an interval order then $Cp = \{(i, j)\}$ where $i \geq j$: if $P_{\leq\varphi} \cup I_{\leq\varphi}$ is an interval order then $P_{\leq\varphi}.I_{\leq\varphi}.P_{\leq\varphi} \subset P_{\leq\varphi}$ which implies $|Cp_{\leq\varphi}| = 1$.

- **Triangle order:** the proof follows directly from the ones of weak orders and of interval orders.

- **Intransitive order:** the proof follows directly from the transitivity of $P_{\leq\varphi}$.

These propositions give us general representations of structures in the sense that these are also true for intervals having more than 3 points. We can conclude now this section by presenting all the 3-points interval representations for the seven preference structures in table 1.

Preference Structure	$\langle P_{\leq\varphi}, I_{\leq\varphi} \rangle$ interval representation
Weak Orders	$Cp_{\leq(3,3,0)} = \{(3,3)\}$ $Cp_{\leq(3,1,1)} = \{(2,2)\}$ $Cp_{\leq(2,2,2)} = \{(1,1)\}$
Bi-weak Orders	$Cp_{\leq(3,1,0)} = \{(2,2), (3,3)\}$ $Cp_{\leq(2,1,1)} = \{(1,1), (2,2)\}$ $Cp_{\leq(2,2,0)} = \{(1,1), (3,3)\}$
Three-Weak Orders	$Cp_{\leq(2,1,0)} = \{(1,1), (2,2), (3,3)\}$
Interval Orders	$Cp_{\leq(0,0,0)} = \{(3,1)\}$ $Cp_{\leq(3,0,0)} = \{(3,2)\}$ $Cp_{\leq(1,1,1)} = \{(2,1)\}$
Split Interval Orders	$Cp_{\leq(1,0,0)} = \{(3,2), (2,1)\}$
Triangle Orders	$Cp_{\leq(1,1,0)} = \{(2,1), (3,3)\}$ $Cp_{\leq(2,0,0)} = \{(1,1), (3,2)\}$
Intransitive Orders	$Cp_{\leq(3,2,0)} = \{(3,3), (1,2)\}$ $Cp_{\leq(2,2,1)} = \{(1,1), (2,3)\}$

Table 1. Preference structures with 3-points interval representation

4 Conclusion

Our study provides an exhaustive view of the comparison of 3-points intervals. Concerning the comparison problem, we have defined and analyzed all the possible 3-points interval comparison procedures that satisfy our axiomatization. Our analysis allows us to know which procedure provides which preference structure and to detect the properties of the resulted preference relations. Concerning the numerical representation problem, we know now all the preference structures having a 3-points interval representation. When ordered points are used, the use of exactly three points is optimal for three-weak orders, triangle orders, split interval orders and intransitive orders, however weak orders, bi-weak orders and interval orders need less than three ordered points. The classical representation of triangle orders and intransitive orders make use of geometric figures (dimension two). As a result they have more complicated comparison rules (for example the comparison of circles is done by a quadratic function) and the representation of preferences needs more space. By proposing a 3-points interval representation we facilitate the comparison rules and the preference representation. We show also that preference structures may have more than one representation.

A possible interesting extension would be the generalization of our study in the case n -points intervals. Such a generalization would offer a general framework for the comparison of ordinal intervals and would allow a systematic study of all the preference structures having an interval representation (for instance for $n=4$ there are 56 sets of P , some of them are already known thanks to the propositions that we presented in this paper).

ACKNOWLEDGEMENTS

This work has been supported by the ANR project PHAC which is gratefully acknowledged. We would like to thank Bruno Escoffier and Olivier Spanjaard for many fruitful discussions on this paper and three anonymous referees for their useful comments.

REFERENCES

- [1] M. Abbas, *Contribution au rapprochement de la théorie des graphes et de l'aide à la décision: graphes parfaits et modèles de préférence*, Ph.D. dissertation, Université Libre de Bruxelles, 1993-1994.
- [2] J.F. Allen, 'Maintaining knowledge about temporal intervals', *Journal of the ACM*, **26**, 832-843, (1983).
- [3] W.E. Armstrong, 'The determinateness of the utility function', *The Economic Journal*, **49**, 453-467, (1939).
- [4] C. Boutilier, 'Toward a logic for qualitative decision theory', in *Proceedings of the 4th International Conference on Knowledge Representation and Reasoning, KR'94*, pp. 75-86. Morgan Kaufmann, San Francisco, (1994).
- [5] R.I. Brafman and M. Tennenholtz, 'On the axiomatization of qualitative decision criteria', in *AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, (1997).
- [6] G. Debreu, *Theory of Value: An Axiomatic Analysis of Economic Equilibrium*, John Wiley and Sons Inc., New York, 1959.
- [7] J. Doyle, 'Prospects for preferences', *Computational Intelligence*, **20**(2), 111-136, (2004).
- [8] P.C. Fishburn, *Interval Orders and Interval Graphs*, J. Wiley, New York, 1985.
- [9] P.C. Fishburn, 'Generalisations of semiorders: a review note', *Journal of Mathematical Psychology*, **41**, 357-366, (1997).
- [10] P. C. Fishburn and W. T. Trotter, 'Split semiorders', *Discrete Mathematics*, **195**, 111-126, (1999).
- [11] R.C. Jeffrey, *The Logic of Decision*, University of Chicago Press, Chicago, 1983.
- [12] R.D. Luce, 'Semiorders and a theory of utility discrimination', *Econometrica*, **24**, 178-191, (1956).
- [13] R.D. Luce and H. Raiffa, *Games and Decisions*, Wiley, New York, 1957.
- [14] M. Ozturk, *Mathematical and logical structures for interval comparisons*, Ph.D. dissertation, Université Paris Dauphine, 2005.
- [15] M. Ozturk, A. Tsoukiàs, and Ph. Vincke, 'Preference modelling', in *Multiple Criteria Decision Analysis: State of the Art Surveys*, eds., M. Ehrgott, S. Greco, and J. Figueira, 27-73, Springer, (2005).
- [16] M. Pirlot and Ph. Vincke, *Semi Orders*, Kluwer Academic, Dordrecht, 1997.
- [17] M. Roubens and Ph. Vincke, *Preference Modelling*, LNEMS 250, Springer Verlag, Berlin, 1985.
- [18] B. Roy, *Méthodologie multicritère d'aide à la décision*, Economica, Paris, 1985.
- [19] S. Russel and P. Norvig, *Artificial Intelligence: a modern approach*, Prentice Hall, New York, 1995.
- [20] W.T. Trotter, *Combinatorics and partially ordered sets: Dimension theory*, John Hopkins University Press, Baltimore, 1992.
- [21] Ph. Vincke, 'Preferences and numbers', in *A-MCD-A - Aide Multi Critère à la Décision - Multiple Criteria Decision Aiding*, eds., A. Colomni, M. Paruccini, and B. Roy, Joint Research Center, 343-354, The European Commission, (2001).

Reference-dependent Qualitative Models for Decision Making under Uncertainty

Patrice Perny and Antoine Rolland¹

Abstract. The aim of this paper is to introduce and investigate a new family of purely qualitative models for decision making under uncertainty. Such models do not require any numerical representation and rely only on the definition of a preference relation over consequences and a relative likelihood relation on the set of events. Within this family, we focus on decision rules using reference levels in the comparison of acts. We investigate both the descriptive potential of such rules and their axiomatic foundations. We introduce in a Savage-like framework, a new axiom requiring that the Decision Maker's preference between two acts depends on the respective positions of their consequences relatively to reference levels. Under this assumption we determine the only possible form of the decision rule and characterize some particular instances of this rule under transitivity constraints.

1 INTRODUCTION

In the past decade, Decision Making under uncertainty has received much attention in Artificial Intelligence. This is also a traditional topic in Economics where several important results have been published, justifying the use of various criteria, for the comparison of actions under risk and uncertainty. The multiplicity of studies and models developed is easily explainable by the variability of contexts for decision making and the diversity of decision making behaviors we want to describe, explain or simulate. In Decision Theory, axiomatic analysis began with the seminal work of Von Neumann and Morgenstern [16] and Savage [17] providing the foundations of the expected utility model (EU). Despite its normative appeal, the descriptive limits of EU have led mathematical economists to study alternatives to Savage's theory and to propose various sophisticated extensions of EU, e.g non-additive integrals like CEU [18]. In this direction, most decision models rely on a cardinal numerical representation of preference (utilities) and uncertainty (probabilities, belief functions and other capacities). Besides this very active current, a parallel stream, with a different focus, is presently developing in AI. Indeed, the need of expressive languages for preference handling and reasoning as well as the aim of developing autonomous decision-making agents in real contexts (with poor information) has led researchers to investigate qualitative models. Although less ambitious in their descriptive and prescriptive objectives, they are more operational in practice because they require less information about preference and likelihood of events. In this direction, several qualitative alternatives to EU have been investigated [2, 3, 7, 15, 8, 13, 14, 19].

As pointed out in [4], most of these models, quantitative or qualitative, rely on a numerical representation where utility and uncertainty are commensurate. To escape this assumption which is not always

seen as natural, recent papers (see e.g. [5, 4]) investigate an alternative approach based on purely qualitative models, i.e. models that only require the definition of a preference order over consequences and a relative likelihood relation on the set of events. This is the case of Likely Dominance Rules [6, 4] that have been recently introduced in AI and characterized by an Ordinal Invariance Axiom (OI) stating that preferences between two acts only depend on the relative position of their consequences for each state. Unfortunately such models do not offer much flexibility to describe beliefs and preferences in practice. On one hand, due to OI they satisfy (as EU) Savage's "Sure Thing Principle" which is sometimes contradicted by observed decision making behaviors. On the other hand, the needs to compose purely ordinal and non-commensurate information (likelihood of events, preferences over consequences) induces a conflict between descriptive and normative objectives. Either preferences rely on a coarse belief structure induced by the existence of a predominant event masking any other event in the analysis or they violate minimal transitivity properties required to justify choices or rankings over acts (this is a consequence of Arrow's theorem, see [1, 4]). To overcome these descriptive and prescriptive limitations, we investigate in this paper an alternative framework (escaping OI) for purely qualitative models. It is based on the introduction of an axiom requiring that preferences between acts depend on the respective position of their consequences (for each state) relatively to reference levels.

The paper is organized as follows: in Section 2, we provide various simple examples showing the descriptive potential of qualitative models based on reference levels. Then, we introduce in Section 3 an axiomatic framework for reference-dependent qualitative models; we determine the possible forms of admissible decision rules. Finally we characterize in Section 4 some particular instance of these rules and discuss the results in the light of Arrow's theorem and extensions.

2 REFERENCE-DEPENDENT PREFERENCES

2.1 Preliminary definitions and notations

We consider here decision-making problems under uncertainty characterized by a 4-tuple $(S, X, \mathcal{A}, \succsim)$, where S is a finite set of the possible *states of nature*, X is the set of the possible *consequences* of acts, $\mathcal{A} = X^S$ is the set of *potential acts*, that is the set of functions $f : S \rightarrow X$, and \succsim is a *preference relation* on \mathcal{A} . Within \mathcal{A} there exist acts leading to the same consequence for all states in S . These acts are called *constant acts* and denoted f_x for any $x \in X$. For any preference relation \succsim , its asymmetric part defines a strict preference denoted \succ and defined by: $x \succ y \Leftrightarrow (x \succsim y \text{ and not}(y \succsim x))$; its symmetric part defines an indifference relation denoted \sim and defined by $x \sim y \Leftrightarrow (x \succsim y \text{ and } y \succsim x)$. If \succsim is transitive, then

¹ LIP6 – University of Paris 6, France, email: firstname.name@lip6.fr

relations \succ and \sim are transitive. A relation \succsim is said to be *quasi-transitive* if its asymmetric part \succ is transitive.

Following Savage terminology, we call *event* any subset A of S . For any pair of acts f and g , the act fAg is defined by: $fAg(s) = f(s)$ if $s \in A$ and $g(s)$ if $s \notin A$. To simplify notation, whenever f or g is a constant act f_x we write fAx (resp. xAg) instead of fAf_x (resp. f_xAg). An event A is said to be null if and only if: $\forall f, g, h \in \mathcal{A} : fAh \sim gAh$. Any event $A \subseteq S$ able to make a discrimination for at least one pair of acts is not null.

2.2 Motivating examples

In order to introduce, by example, likelihood dominance rules as well as reference-dependent models, consider the following game:

Example 1 Consider a two-player game with 3 fair dice with unusual points (see [12]) $f = (1, 4, 4, 4, 4, 4)$, $g = (3, 3, 3, 3, 3, 6)$ and $h = (2, 2, 2, 5, 5, 5)$. Player 1 gets a choice of whichever die he wants, then Player 2 picks one of the remaining dice. They each roll n times, and whoever scores higher the most times, wins. As the second player, which die should we pick?

This can be formalized by considering 8 (virtual) states corresponding to all possible combinations of outcomes, as follows:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
P	5/72	5/72	1/72	1/72	25/72	25/72	5/72	5/72
f	1	1	1	1	4	4	4	4
g	3	3	6	6	3	3	6	6
h	2	5	2	5	2	5	2	5

Here preferences over dice can be stated by a Likely Dominance Rule (LDR, [4]): $f \succsim g \Leftrightarrow \{s \in S, f(s) \geq g(s)\} \succsim_{\Lambda} \{s \in S, g(s) \geq f(s)\}$ where \succsim_{Λ} is a likelihood relation on the sets of events $\Lambda = 2^S$ induced by the probability of events. Such a definition only uses two binary relations (\geq and \succsim_{Λ}). Note that we have here: $P(f > g) = 50/72$, $P(g > h) = 42/72$, $P(h > f) = 42/72$, which means that $f \succ g$, $g \succ h$ and $h \succ f$ yielding to intransitive preferences. This gives a winning strategy to Player 2 who can always choose a better die than Player 1. The possibility of describing intransitive preferences is typical of Likely Dominance Rule as defined in [4] to compare acts in decision making problems under uncertainty. A variant of the game consists in changing the rule into: “whoever scores higher or equal than ‘4’ the most times, wins”. Here preferences over dice are defined by: $f \succsim g \Leftrightarrow \{s \in S, f(s) \geq 4\} \succsim_{\Lambda} \{s \in S, g(s) \geq 4\}$. In this case, the introduction of reference level ‘4’ yields a transitive preference order: $f \succ h \succ g$. Such a model is an example of reference-depend LDR we intend to investigate in the paper.

Example 2 We consider the “Ellsberg” example [9] with an urn containing blue, red, and green balls. We only know that the proportion of blue balls is 1/3. The player is invited to pay \$p to play one of the following games f and g : he draws a ball then f wins \$ $q > 3p$ if the ball is blue and g wins \$ q if the ball is red. Then he has a second choice between two other games: f' wins \$ q if the ball is blue or green, g' wins \$ q if the ball is red or green.

Experiments show that people often prefer f to g (because they are not sure that there is at least 1/3 of red balls in the urn) but prefer g' to f' (because they are sure there is 2/3 of red or green balls). Such a decision problem can be formalized as follows: We consider 3 states s_1 (blue), s_2 (red), s_3 (green) and the two possible consequences $x = q - p$, $y = -p$. The 4 games are represented by acts f, g, f', g' with the following consequences:

	f	g	f'	g'
s_1	x	y	x	y
s_2	y	x	y	x
s_3	y	y	x	x

Such preferences are not compatible with the so-called “Sure-thing Principle” (P2) introduced by Savage [17] in the characterization of EU. This principle states that the preference $f \succsim g$ does not depend on states where the two acts have the same consequences. Here this principle is violated because we have $f \succ g$ and $g' \succ f'$, a strict preference reversal only due to a shift from y to x on s_3 . This reversal makes impossible to represent such preferences by EU or by its qualitative counterparts proposed in [7]. Remark that $\{s \in S, f(s) \geq g(s)\} = \{s \in S, f'(s) \geq g'(s)\} = \{s_1, s_3\}$ and $\{s \in S, g(s) \geq f(s)\} = \{s \in S, g'(s) \geq f'(s)\} = \{s_2, s_3\}$. Then the different status given to pairs (f, g) and (f', g') show that preferences between two acts do not only depend on the relative position of their consequences for each state. This makes it impossible to represent such preferences with an LDR. However, the introduction of a reference level can easily solve the problem. Indeed preferences can be explained by the following reference-dependent rule:

$$f \succsim g \iff \{s \in S, f(s) \geq 0\} \succsim_{\Lambda} \{s \in S, g(s) \geq 0\} \quad (1)$$

with $s_1 \succsim_{\Lambda} s_2$ and $\{s_2, s_3\} \succsim_{\Lambda} \{s_1, s_3\}$. These examples show that the introduction of one (or several) reference levels can facilitate the description of observed preferences in some cases. We investigate now in a more formal framework reference-depend ordinal models.

3 AN AXIOMATIC FRAMEWORK

3.1 Axioms

We introduce in this section some axioms for reference-dependent preferences. We start from axioms considered by Savage [17] in the characterization of the EU model and introduce some variations more suitable in our framework. Then we will introduce more specific axioms designed to capture reference-dependent preferences. In the original framework, Savage assumes that Decision Maker’s (DM) preferences over acts have a complete weak order structure (P1 principle). This is necessary to characterize the EU model which requires, by definition, completeness and transitivity of preferences.

Axiom P1. \succsim is a complete weak-order on \mathcal{A} , i.e., \succsim is reflexive, complete and transitive.

We consider here a wider class of models including partial preference structures not necessarily transitive. Hence, we recall the P1 principle of Savage and introduce a weak version relaxing transitivity for quasi-transitivity:

Axiom WP1. \succsim is reflexive and quasi-transitive and its restriction to constant acts is a complete order.

The second axiom of Savage is the so-called *sure-thing principle* (P2). As mentioned above, it requires that the preference between two acts fAh and gAh does not depend on the choice of h . This axiom does not necessarily hold when reference levels are used in a decision models. This is quite natural as reference-dependent preferences might be such that $fAh \succ gAh$ and $gAh' \succ fAh'$, as shown by Example 2. In this example indeed, setting $A = \{s_1, s_2\}$ we have $f = fAy$, $g = gAy$, $f' = fAx$, and $g' = gAx$. Hence $f \succ g$ and $g' \succ f'$ contradict P2. This strict preference reversal would also contradict a weaker version of P2 considered in the axiomatization of qualitative utilities [8].

In order to reveal Decision Maker's preferences over certain consequences, the comparison of constant acts is of particular interest. We can indeed consider the following relation:

$$\forall x, y \in X, (x \succsim_X y \Leftrightarrow f_x \succsim f_y) \quad (2)$$

This definition is self consistent provided that the preference over constant acts should not admit rank reversal of type $f_x Ah \succ f_y Ah$ and $f_x Ah' \prec f_y Ah'$. This can be captured by the following Savage's P3 postulate:

$$\text{Axiom P3. } \forall A \subseteq S, \forall h \in A, (x \succsim_X y \Leftrightarrow f_x Ah \succsim f_y Ah).$$

However, $x \succ_X y$ is not always sufficient to justify a preference of type $f_x Ah \succ f_y Ah$. For example, when A is a weakly plausible event, we might have: $f_x Ah \sim f_y Ah$ for some h . This justifies the following weakening of P3:

$$\text{Axiom WP3. } \forall A \subseteq S, \forall h \in A, (x \succ_X y \Rightarrow f_x Ah \succsim f_y Ah).$$

The fourth principle P4 is introduced by Savage to reveal from \succsim a consistent likelihood relation over the set of events. This axiom writes as follows:

$$\text{Axiom P4. } \forall A, B \subseteq S, \forall x, y, x', y' \in X : x \succ_X y \text{ and } x' \succ_X y', xAy \succsim xBy \Leftrightarrow x'Ay' \succsim x'By'.$$

We will not directly use P4 here, as it is a by-product of a stronger axiom introduced later. We introduce now the P5 principle of Savage requiring that preferences over constant acts are not trivial.

$$\text{Axiom P5. } \exists x, y \in X \text{ s.t. } f_x \succ f_y \text{ (i.e. } x \succ_X y\text{).}$$

Under WP1 and P5 we know that \sim_X , the symmetric part of \succsim_X is an equivalence relation with at least two classes. Assuming there exist $q+1$ distinct equivalence classes ($1 \leq q < |X|$), r_i will denote any element of X representing class i , in such a way that $f_{r_i} \succ_X f_{r_{i+1}}$, $i = 1, \dots, q$. The set of all but the worst reference levels is denoted $\mathcal{R} = \{r_1, \dots, r_q\}$ and represents the various significant separating levels in X . In the next section, to avoid trivial cases, we will consider SP5, a slight reinforcement of P5 requiring 4 distincts preference levels on the consequence scale. This means that at least 3 different reference levels do exist, which implies that the structure is rich enough to allow problematic situations as Condorcet triples (see Example 5). The axiom writes as follows:

$$\text{Axiom SP5. } \exists x, y, z, w \in X \text{ s.t. } f_x \succ f_y \succ f_z \succ f_w.$$

We introduce now a more specific axiom stating that preferences over acts only depend on the respective positions of their consequences relatively to these *reference levels*. To this end, the following *reference events* have a particular importance:

$$\forall r \in \mathcal{R}, F_r = \{s \in S, f(s) \succsim_X r\} \quad (3)$$

For acts g, h, \dots we will use the notations G_r, H_r, \dots respectively. For the sake of simplicity, event F_{r_k} will be denoted F_k , for $k = 1, \dots, q$. With these notations, the axiom writes:

$$\text{Axiom SRL (Separability w.r.t. Reference Levels) } \forall f, g, f', g' \in \mathcal{A}, \forall i, j \in \{1, \dots, q\},$$

$$\left. \begin{array}{l} F_i = F'_j, G_i = G'_j \\ \forall k \neq i, F_k = G_k \\ \forall k \neq j, F'_k = G'_k \end{array} \right\} \Rightarrow [f \succsim g \Leftrightarrow f' \succsim g']$$

A direct consequence of SRL is the following:

$$\text{Axiom DRE (Dependence w.r.t. Reference Event)}$$

$$\left. \begin{array}{l} \forall r \in \mathcal{R}, F_r = F'_r \\ \forall r \in \mathcal{R}, G_r = G'_r \end{array} \right\} \Rightarrow [f \succsim g \Leftrightarrow f' \succsim g']$$

Axiom DRE requires that the preference between two acts only depends on the respective position of their consequences (for each state) with respect to reference levels in \mathcal{R} . In this respect, it can be seen as the natural variant of the Ordinal Invariance Axiom [4] enforcing ordinal comparisons between acts to rely on reference levels. It can easily be shown that SRL implies DRE, but the converse is not true as can be seen in the following example.

Example 3 Suppose that $S = \{s_1, s_2, s_3, s_4\}$ and consider x, y, z 3 consequences of X such that $x \succ_X y \succ_X z$. Consider two reference levels $r_1 = x, r_2 = y$ and 4 acts $f, g, f', g' \in \mathcal{A}$ such that:

	s_1	s_2	s_3	s_4		s_1	s_2	s_3	s_4	
f	x	y	y	z		f'	y	x	y	z
g	x	y	z	y		g'	y	x	z	y

We have $F_1 = G_1 = \{s_1\}$, $F'_1 = G'_1 = \{s_2\}$, $F_2 = F'_2 = \{s_1, s_2, s_3\}$ and $G_2 = G'_2 = \{s_1, s_2, s_4\}$. If $f \succ g$ and $g' \succ f'$, then axiom SRL does not hold (just choose $q = 2, i = j = 2, k = 1$ in the formulation of SRL). Yet this situation is compatible with DRE.

In the context of reference-dependent models, the relative likelihood of events can be assessed by observing conflicting events in the respective comparison of consequences of two acts f and g relatively to reference levels. Following this idea, a relative likelihood relation \succsim_Λ over the set of events $\Lambda = 2^S$ might be derived from \succsim by setting, for all $A, B \subseteq S$:

$$A \succsim_\Lambda B \Leftrightarrow \exists f, g \in \mathcal{A}, \exists r \in \mathcal{R}, \left\{ \begin{array}{l} F_r = A, G_r = B \\ \forall t \in \mathcal{R} - \{r\}, F_t = G_t \\ f \succsim g \end{array} \right. \quad (4)$$

Such a construction only makes sense if there is no antagonist pairs of acts (f, g) and (f', g') inducing respectively $A \succsim_\Lambda B$ and $B \succsim_\Lambda A$ by Equation (4). Axiom SRL excludes such situations: suppose that two subsets of S , A and B are such that there exists $f, g \in X$, $f', g' \in X$ such that $f \succsim g$ implies $A \succsim_\Lambda B$ and $g' \succ f'$ implies $B \succsim_\Lambda A$. If $f \succsim g$ implies $A \succsim_\Lambda B$, it means that $\exists r \in \mathcal{R}, F_r = A$, $G_r = B$ and $\forall p \neq r, F_p = G_p$. If $g' \succ f'$ implies $B \succsim_\Lambda A$, it means that $\exists r' \in \mathcal{R}, F'_{r'} = A, G'_{r'} = B$ and $\forall p \neq r', F'_p = G'_p$. Such a situation is not compatible with the respect of axiom SRL.

3.2 Ordinal reference dependent rules

Under SRL we can define, for any reference level $r \in \mathcal{R}$ a reference-based preference relation \succsim_r over acts by:

$$f \succsim_r g \Leftrightarrow F_r \succsim_\Lambda G_r \quad (5)$$

The profile $(\succsim_{r_1}, \dots, \succsim_{r_q})$ of reference-based preferences relations contains any useful information that might be used in an ordinal reference dependent model. We introduce now an axiom of *ordinal invariance with respect to reference-based preferences* (OIRP) requiring that preferences between two acts only depend on the relative position of these acts in the q reference-based preferences relations.

$$\text{Axiom OIRP } \forall f, g, f', g' \in \mathcal{A}:$$

$$\left. \begin{array}{l} \forall r \in \mathcal{R}, f \succsim_r g \Leftrightarrow f' \succsim_r g' \\ \forall r \in \mathcal{R}, g \succsim_r f \Leftrightarrow g' \succsim_r f' \end{array} \right\} \Rightarrow [f \succsim g \Leftrightarrow f' \succsim g']$$

This axiom is a transposition to reference-based preferences of the OI axiom used in [4] to characterize Likely Dominance Rules. Without OIRP, nothing guarantees that the preference between two acts f and g only depends on the status of relations $\succsim_{r_j}, j = 1, \dots, q$ on the pair (f, g) .

Example 4 Suppose that x, y, z, w are four consequences of X such that $x \succ_X y \succ_X z \succ_X w$. Let x, y and z be three reference levels and f, g, f', g' be four acts of \mathcal{A} such that:

$s_1 \quad s_2 \quad s_3$			$s_1 \quad s_2 \quad s_3$		
$f \quad x \quad y \quad w$	$f' \quad x \quad w \quad w$		$g \quad y \quad y \quad z$	$g' \quad y \quad z \quad z$	

Suppose that the DM beliefs about possible events are such that $A \succsim_\Lambda B \Leftrightarrow |A| \geq |B|$ (which means that, for example, $(x, y, y) \succ (y, y, y)$). Then we have $f \succ_x g$, $f \sim_y g$, $f \prec_z g$ and $f' \succ_x g'$, $f' \sim_y g'$, $f' \prec_z g'$. If DM preferences are such that $f \succ g$ and $g' \succ f'$, which does not contradict axiom SRL, then nothing in the profile $(\succsim_x, \succsim_y, \succsim_z)$ can explain the different status of pairs (f, g) and (f', g') according to \succsim . OIRP precisely avoids such preference situations. Actually it avoids any situation where a preference between two acts might depend on other acts (an idea closed to Arrow independence of irrelevant alternative [1]).

Hence SRL and OIRP characterize an interesting class of reference-dependent preferences that can be obtained by ordinal aggregation of the profile $(\succsim_{r_1}, \dots, \succsim_{r_q})$. Among others, one can check that the combination of SRL and OIRP implies P4. We can now state a proposition giving the general form of such reference-based likely dominance rules (RLD rules for short). Note that the following proposition is to be interpreted under the WP3 assumption, although not formally necessary to establish the result.

Proposition 1 If the preference relation \succsim on \mathcal{A} satisfies axioms SRL and OIRP, then there exists a relation \succsim_R defined on $R = 2^{\mathcal{R}}$ such that:

$$f \succsim g \Leftrightarrow \{r \in \mathcal{R}, f \succsim_r g\} \succsim_R \{r \in \mathcal{R}, g \succsim_r f\} \quad (6)$$

Proof. Thanks to SRL, an importance relation \succsim_Λ can soundly be revealed from \succsim by Equation (4). We then define a relation \succsim_R on the subsets of \mathcal{R} as follows: consider Q and Q' two subsets of \mathcal{R} , $Q \succsim_R Q' \Leftrightarrow \exists f, g \in \mathcal{A}$ such that $[Q = \{r \in \mathcal{R}, F_r \succsim_\Lambda G_r\}, Q' = \{r \in \mathcal{R}, G_r \succsim_\Lambda F_r\}]$ and $f \succsim g$. Hence, suppose that there are two couples of $\mathcal{A} \times \mathcal{A}$ f, g and f', g' such that $Q = \{r \in \mathcal{R}, F_r \succsim_\Lambda G_r\} = \{r \in \mathcal{R}, F'_r \succsim_\Lambda G'_r\}$ and $Q' = \{r \in \mathcal{R}, G_r \succsim_\Lambda F_r\} = \{r \in \mathcal{R}, G'_r \succsim_\Lambda F'_r\}$. Thanks to axiom OIRP, we have $f \succsim g \Leftrightarrow f' \succsim g'$, which showss that relation \succsim_R is soundly defined. Indeed, we cannot have $Q \succsim_R Q'$ with a pair (f, g) and $Q \prec_R Q'$ with another pair (f', g') . \square

Relation \succsim_R reflects the relative importance attached to subsets of consequences by the DM. The reference-based rule given in Equation (6) is formally equivalent to a Likely Dominance Rule [4] applied on reference-dependent relations \succsim_r for all $r \in \mathcal{R}$ instead of state-dependent relations \succsim_s defined by $f \succsim_s g \Leftrightarrow f(s) \succsim_X g(s)$, for all $s \in S$. In the next section, we investigate the structure of such rules under the assumption of transitivity for \succsim .

4 AXIOMATIC CHARACTERIZATION

The axioms used to establish Proposition 1 are not sufficient to derive a fully operational decision rule. Indeed, nothing guarantees that \succsim constructed according to Equation (6) is transitive or even quasi-transitive. The following example shows indeed that natural relations \succsim_R based on a majority rule might fail to produce transitive relations, as Condorcet's triplets can appear in the profile $(\succsim_{r_1}, \dots, \succsim_{r_q})$:

Example 5 Under SP5, we can assume that there exist $r_1, r_2, r_3 \in \mathcal{R}, x, y, z, w \in X$ so that $x \succsim_X r_1 \succ_X y \succsim_X r_2 \succ_X z \succsim_X r_3 \succ_X w$. Let \succsim_Λ be such that $\{s_1, s_2, s_3\} \succsim_\Lambda \{s_1, s_2\} \succsim_\Lambda \{s_1, s_3\} \succsim_\Lambda \{s_2, s_3\}$ and $f = (x, w, z), g = (z, x, z)$ and $h = (y, y, w)$. Then we have $\begin{cases} F_1 = \{s_1\} & F_2 = \{s_1\} & F_3 = \{s_1, s_3\} \\ G_1 = \{s_2\} & G_2 = \{s_2\} & G_3 = \{s_1, s_2, s_3\} \\ H_1 = \emptyset & H_2 = \{s_1, s_2\} & H_3 = \{s_1, s_2\} \end{cases}$ which implies that $f \succsim_1 g \succsim_1 h, h \succsim_2 f \succsim_2 g$ and $g \succsim_3 h \succsim_3 f$

Note that we get a profile forming a *Condorcet triple* which would induce $f \succ g, g \succ h$ and $h \succ f$ if the importance relation \succsim_R is such that: $Q \succsim_R Q' \Leftrightarrow |Q| \geq |Q'|$. Such a preference would not be very helpful for decision making due to the existence of preference cycles. For this reason, we investigate now the impact of transitivity requirements on \succsim in our framework.

Under SRL and OIRP assumptions, if P1 and WP3 hold then nice properties of the importance relation \succsim_Λ appear. A first one is that \succsim_Λ is a transitive relation, which derives from Equation (4) and transitivity of \succsim . A second one is monotonicity defined as follow:

Axiom MON (monotonicity w.r.t. set inclusion). Preference relation \succsim , satisfying axioms SRL and OIRP is said to be monotonic if $\forall A, B, C, D \subseteq S$, we have:

$$[A \subseteq B \text{ and } C \subseteq D] \Rightarrow [A \succsim_S D \Rightarrow B \succsim_S C]$$

Proposition 2 Under SRL and OIRP, (P1 and WP3) \Rightarrow MON

Proof: We first establish the following:

Lemma: let $r, r' \in \mathcal{R}$ such that $r \succ r', f = rAr', g = rBr'$. Then $A \subseteq B \Leftrightarrow g \succsim f$.

Proof of the lemma: Suppose that $A \subseteq B$. WP3 gives immediately that $rB \cap \bar{A}f \succsim r'B \cap \bar{A}f$, and as $g = rB \cap \bar{A}f$ and $f = r'B \cap \bar{A}f$, it gives immediately that $g \succsim f$.

We prove now the proposition. Let $A, B, C, D \subseteq S$ such that $A \subseteq B$ and $C \subseteq D$. Let $r_k, r_{k+1} \in \mathcal{R}$ such that $r_1 \succ_X \dots \succ_X r_k \succ_X r_{k+1} \succ_X \dots \succ_X r_q$. Let $f, g, f', g' \in \mathcal{A}$ such that: $f = r_k A r_{k+1}$, $g = r_k B r_{k+1}$, $f' = r_k C r_{k+1}$, $g' = r_k D r_{k+1}$. If $f \succsim g'$ we have $A \succsim_\Lambda D$ by definition of \succsim_Λ . Then, the lemma gives $B \succsim_\Lambda A$, and so $B \succsim_\Lambda D$ by transitivity of \succsim_Λ . Finally, by contraposition of the lemma, we have $B \succsim_\Lambda C$, which proves MON. \square

We want now to characterize the structure of decision rules defining, from any profile $(\succsim_{r_1}, \dots, \succsim_{r_q})$, a transitive preference relation \succsim on \mathcal{A} using Equation (6) while satisfying the following axiom named IRP for *Independence for Reference-based Preferences*:

Axiom IRP For all \succsim and \succsim' inducing reference sets \mathcal{R} and \mathcal{R}' of same cardinality, $\forall f, g, f', g' \in \mathcal{A}$:

$$\forall r \in \mathcal{R}, \forall r' \in \mathcal{R}', \quad \begin{cases} f \succsim_r g \Leftrightarrow f' \succsim'_{r'} g' \\ g \succsim_r f \Leftrightarrow g' \succsim'_{r'} f' \end{cases} \Rightarrow [f \succsim g \Leftrightarrow f' \succsim' g']$$

We can now establish the following representation theorem:

Theorem 1 If the preference relation \succsim on \mathcal{A} satisfies P1, WP3, SP5, SRL and IRP then there exists a permutation σ on $\{1, \dots, q\}$, and a likelihood relation \succsim_Λ on the subsets of S such that:

$$\begin{aligned} f \succ g &\Leftrightarrow F_{\sigma(1)} \succsim_\Lambda G_{\sigma(1)} \\ &\text{or} \quad F_{\sigma(1)} \sim_\Lambda G_{\sigma(1)} \text{ and } F_{\sigma(2)} \succsim_\Lambda G_{\sigma(2)} \\ &\dots \\ &\text{or} \quad \forall i \neq q, F_{\sigma(i)} \sim_L G_{\sigma(i)} \text{ and } F_{\sigma(q)} \succsim_\Lambda G_{\sigma(q)} \\ f \sim g &\Leftrightarrow \forall i = 1, \dots, q, F_i \sim_\Lambda G_i \end{aligned}$$

Sketch of the proof. Under P1 and SP5, we know there exists at least $q \geq 3$ reference levels in $\mathcal{R} = \{r_1, \dots, r_q\}$ so that $f_{r_1} \succ \dots \succ f_{r_q}$. Then, the proof proceeds as in the Fishburn's characterization result for lexicographic preferences [11], translated from a product

set to Arrow's original framework [1]. Before, we have to prove that profiles $(\succsim_{r_1}, \dots, \succsim_{r_q})$ have the adequate structure. To this end, we first show that \succsim_R satisfies the strong Pareto principle, i.e. $(\forall r \in \mathcal{R}, f \succ_r g) \Rightarrow f \succ g$. This is due to the following property: $\forall r \in \mathcal{R} - \{r_i\}$, $r_i \succsim_r r_{i+1}$, and $r_i \succ r_{i+1}$ by definition of reference levels and $r_i \succsim_{r_i} r_{i+1}$ since $S \succsim_\Delta \emptyset$ thanks to P5 (actually SP5). So $\forall f, g \in \mathcal{A}, [\forall r \in \mathcal{R}, f \succsim_r g \text{ and } \exists r \in \mathcal{R}, f \succ_r g]$ implies $f \succ g$ which proves the strong Pareto principle. Then our IRP axiom plays the role of Fishburn's non-compensation axiom on product sets. Finally the richness of the product set structure exploited in Fishburn's result is obtained here by the diversity of relationship between acts due to the existence of Condorcet triples illustrated by Example 5. \square

Theorem 1 shows that, under IRP and the transitivity requirement (P1), RLD rules necessarily perform a lexicographic aggregation of relations \succsim_{r_i} , $i = 1, \dots, q$, thus inducing the existence of a 'dictator' among reference levels (a transposition of Arrow's theorem), and even the existence of a hierarchy of reference levels. Reference r_i is only allowed to discriminate between two acts f and g when reference levels r_k , $k < i$ (higher in the hierarchy) are useless in the comparing f and g . The hierarchy order is completely characterized by permutation σ on $\{1, \dots, q\}$ and can easily be revealed by observing preferences between $r_{j+1}Ar_{j-1}$ and r_j for any j where A is such that $N \succsim_\Delta A \succsim_\Delta \emptyset$. Indeed, $r_{j+1}Ar_{j-1} \succ r_j$ implies that r_{j+1} is above r_j in the hierarchy and vice versa. A very natural example of RLD rule is given by choosing $\sigma(k) = q+1-k$. It amounts to comparing two acts f and g by observing their respective potential to yield consequences better than a minimal reference level. If this is not sufficient to discriminate them, then a second (more demanding) level is considered and so on...

The key point with such aggregation procedures is that the hierachic structure does not apply on *states* (as is the case for usual LDR rules induced by axiom OI, see [4]) but on *consequences*. This leaves room for various types of complete monotonic likelihood relations \succsim_Δ , as those induced by capacities (additive or not, decomposable or not). In this respect, this offers much more descriptive possibilities than LDR rules. Moreover, preferences induced by LDR rules are, by construction, compatible with the qualitative counterpart of stochastic dominance \succsim_Δ defined on \mathcal{A} by:

$$f \succsim_\Delta g \Leftrightarrow \forall x \in X, \{s \in S, f(s) \succsim_X x\} \succsim_\Delta \{s \in S, g(s) \succsim_X x\}$$

5 CONCLUSION

We have investigated new qualitative models for decision making under uncertainty, characterized by the introduction of reference levels. The main features of the resulting RLD rules are the following:

1) *level of information required*: models are purely ordinal. The decision rule is completely characterized by two binary relations \succsim_Δ (relative likelihood on events) and \succsim_X (preference over consequences). This departs from the great majority of models based on a numerical qualitative or quantitative decision criterion, the definition of which relies on the existence of certainty equivalent implicitly merging utility and uncertainty scales into a single one [4].

2) *prescriptive potential*: although based on purely ordinal methods, they are fully compatible with the transitivity of preferences (even if a probabilistic likelihood relation is used). The use of reference levels and reference events indeed overcomes the usual limitations of purely ordinal aggregation methods, by moving the application point of Arrow-like theorems. The introduction of SRL instead of OI exchanges the role of events and consequences in the ordinal aggregation. Thus, \succsim is obtained by aggregation of reference-dependent relations $\succsim_r, r \in \mathcal{R}$ instead of state-dependent preferences $\succsim_s, s \in S$.

As an important consequence we get Theorem 1 showing that necessary lexicographic structures induced by transitivity emerge on the space of consequences rather than on the space of events (see LDR [4]). In our opinion, this is much more satisfactory.

3) *Descriptive potential*: many recent studies in decision making under uncertainty concern the violation of the sure-thing principle P2 and the need of models to capture such sophisticated preferences. As Choquet expected utility [18] is a useful extension of EU in the field of quantitative models, as Sugeno Expected Utility [8] is a useful extension of qualitative utility, RLD rules provide a new family of Dominance Rules allowing, if necessary, violations of P2.

As future works, we wish to investigate a weaker version of SRL, leaving room for a set of q possibly different reference-dependent likelihood relations. More crucially, we have to investigate the connections of our approach (under the completeness assumption) with the Sugeno integral and with possibilistic lexicographic decision rules proposed in [10]. Finally, we intend to extend our characterization results under quasi-transitivity and/or without completeness of preferences. This might be a step towards the definition of new qualitative decision models.

REFERENCES

- [1] K. J. Arrow, *Social Choice and Individual Values*, Cowles Foundations and Wiley - New-York, 1951.
- [2] C. Boutilier, 'Toward a logic for qualitative decision theory', in *Proc. of the International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pp. 75–86, (1994).
- [3] R. Brafman and M. Tennenholtz, 'On the foundations of qualitative decision theory', in *Proc. 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 1291–1296, Portland, OR, (1996).
- [4] D. Dubois, H. Fargier, and P. Perny, 'Qualitative decision theory with preference relations and comparative uncertainty: an axiomatic approach', *Artificial Intelligence Journal*, **148**, 219–260, (2003).
- [5] D. Dubois, H. Fargier, P. Perny, and H. Prade, 'Qualitative decision theory: from Savage's axioms to nonmonotonic reasoning', *Journal of the Association of Computer Machinery*, **49**(4), 455–495, (2002).
- [6] D. Dubois, H. Fargier, and H. Prade, 'Decision-making under ordinal preferences and comparative uncertainty', in *Proc. of UAI'97*, eds., D. Geiger and P. P. Shenoy, pp. 157–164, (1997).
- [7] D. Dubois and H. Prade, 'Possibility theory as a basis for qualitative decision theory', in *Proc. of IJCAI'95, Montreal*, pp. 1924–1930, (1995).
- [8] D. Dubois, H. Prade, and R. Sabbadin, 'Qualitative decision theory with Sugeno integrals', in *Proc. of UAI'98*, pp. 121–128, (1998).
- [9] D. Ellsberg, 'Risk, ambiguity and the Savage axioms', *Quarterly Journal of Economics*, **75**, 643–669, (1961).
- [10] H. Fargier and R. Sabbadin, 'Qualitative decision under uncertainty: back to expected utility', *Artificial Intelligence*, **164**, 245–280, (2005).
- [11] P. C. Fishburn, 'Axioms for lexicographic preferences', *Review of Economic Studies*, **42**, 4.5–419, (1975).
- [12] M. Gardner, 'Mathematical games: The paradox of the nontransitive dice and the elusive principle of indifference', *Scientific American*, **223**, 110–114, (1970).
- [13] P. Giang and P. Shenoy, 'A comparison of axiomatic approaches to qualitative decision-making using possibility theory', in *Proc. of 17th Conf. on Uncertainty in Artificial Intelligence, San Francisco*, pp. 162–170. Morgan Kaufmann, (2001).
- [14] L. Godo and A. Zapico, 'Lexicographic refinements in the context of possibilistic decision theory', in *EUSFLAT*, volume 4, (2005).
- [15] D. Lehmann, 'Generalized qualitative probability : Savage revisited', in *Proc. of 12th Conf. on Uncertainty in Artificial Intelligence, Portland, OR*, pp. 381–388, (1996).
- [16] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior*, Princeton University Press, 1947.
- [17] L. J. Savage, *The Foundations of Statistics*, J. Wiley and Sons, New-York, 1954.
- [18] D. Schmeidler, 'Integral representation without additivity', *Proceedings of the American Mathematical Society*, **97**(2), 255–261, (1986).
- [19] P. Weng, 'Qualitative decision making under possibilistic uncertainty: Toward more discriminating criteria', in *UAI*, volume 21, (2005).

Decision with uncertainties, feasibilities, and utilities: towards a unified algebraic framework

Cédric Pralet^{1,3} and Gérard Verfaillie² and Thomas Schiex³

Abstract. Several formalisms exist to express and solve decision problems. Each is designed to capture different kinds of knowledge: utilities expressing preferences, uncertainties on the environment, or feasibility constraints on the decisions, with a possible sequential aspect. Despite the fact that every framework relies on specific properties exploited by dedicated algorithms, these formalisms present interesting similarities.

In this paper, we show that it is possible to capture these similarities in a generic algebraic framework for sequential decision making with uncertainties, feasibilities, and utilities. This framework subsumes several existing approaches, from constraint satisfaction problems to quantified boolean formulas, Bayesian networks or possibilistic Markov decision processes. We introduce this framework using a toy example, increasingly sophisticated by uncertainties, feasibilities and possible observations. This leads to a formal definition of the framework together with dedicated queries representing usual decision problems. Generic algorithms for solving the considered queries should allow to both factorize existing algorithmic works and allow for cross-fertilization between the subsumed formalisms.

1 Introduction and notations

The notion of decision problems covers a large spectrum of problems, from pure utility optimization problems to problems involving uncertainties, possible unfeasibilities and partial observability. A large number of frameworks have therefore been proposed to model and solve such problems. Our aim here is to define a general framework based on graphical models (to capture locality of information and independence) enriched by an algebraic framework allowing to design algorithms and prove properties. All proofs are omitted for lack of space and available in [27].

In the following, $\text{Dom}(x)$ denotes the set of values a variable x may take. By extension, for a sequence of variables S , $\text{Dom}(S) = \prod_{x \in S} \text{Dom}(x)$. Given a set E , a *local function* L is a function $L : \text{Dom}(S(L)) \rightarrow E$, where $S(L)$ is the scope of L . The *elimination of a set of variables* S' from L with any given associative commutative operator op on E is defined as $(op_{S'} L)(A) = op_{A' \in \text{Dom}(S')} L(A.A')$ for any assignment A of $S - S'$. Boolean values are denoted t and f .

We start with a first basic example using just utilities.

Example John faces three doors A, B, C. One of the doors hides a treasure, another a gangster. John can decide to open one door. The gangster will rob him 4,000€ but the treasure is worth 10,000€.

¹ LAAS-CNRS, Toulouse, France cprale@laas.fr

² ONERA, Toulouse, France gerard.verfaillie@onera.fr

³ INRA, Castanet-Tolosan, France tschiex@toulouse.inra.fr

Modeling To compactly represent the environment and the decisions, we introduce three variables: (1) two *environment variables*: one for the gangster door (denoted ga), and one for the treasure door (tr); (2) one *decision variable* (do), representing the door John decides to open. Every variable has $\{A, B, C\}$ as domain. Decision variables are variables whose value is controlled by an agent. Otherwise they are environment variables.

Then, we need two local *utility functions* U_1, U_2 to represent utilities: (1) U_1 expresses that if John opens the gangster door, he must pay 4,000€ (soft constraint $do = ga$, with utility degree $-4,000\text{€}$ if satisfied, and 0 otherwise); (2) U_2 expresses that if John opens the treasure door, he wins 10,000€ (soft constraint $do = tr$, with utility degree $10,000\text{€}$ if satisfied, and 0 otherwise).

Associated query Which door John should open if he knows that the gangster is behind door A and that the treasure is behind door C (no uncertainties)? Obviously, he should open door C.

1.1 Adding uncertainties

In real problems, the environment may not be completely known: there may be uncertainties (here called *plausibilities*) as well as possible *observations* on this uncertain environment. We sophisticate our treasure quest problem to integrate such aspects.

Example The treasure and the gangster are not behind the same door, and all situations are equiprobable. John is accompanied by Peter. Each of them can decide to listen in to door A, B, or C to try to detect the dog of the gangster. The probability of hearing something is 0.8 if one listens in to the gangster door, 0.4 for a door next to the gangster one, and 0 otherwise.

Modeling To capture these new specifications, we define (1) four more variables: two decision variables li_J and li_P , with $\{A, B, C\}$ as domain, model the doors to which John and Peter listen in, and two environment variables he_J and he_P , with $\{yes, no\}$ as domain, model whether John and Peter hear the dog; (2) four local *plausibility functions*: $P_1 : ga \neq tr$ and $P_2 = 1/6$ model probability distribution on the gangster and treasure positions; $P_3 = P(he_J | li_J, ga)$ defines the probability that John hears something given the door to which he listens in and the gangster door; similarly, P_4 corresponds to $P(he_P | li_P, ga)$. Implicitly, the local plausibilities satisfy normalization conditions. First, as the treasure and the gangster are somewhere, $\sum_{ga, tr} P_1 \times P_2 = 1$. Then, as John and Peter hear something or not, $\sum_{he_J} P_3 = 1$ and $\sum_{he_P} P_4 = 1$.

Associated queries What are the decision rules that maximize the expected utility, if first Peter and John listen in, and then John decides to open a door knowing what has been heard?

A classical approach to answer such a query is to use a *decision tree*. In this tree, variables can be considered in the order $li_J \rightarrow li_P \rightarrow he_J \rightarrow he_P \rightarrow do \rightarrow ga \rightarrow tr$ (first, John and Peter choose a door to listen in, then they listen and depending on the listening, John decides which door to open; finally the gangster and the treasure are behind a given door with a certain probability). An internal node n in the tree corresponds to a variable x , and an edge in the tree is labeled with an assignment $x = a$ of the variable x associated with the node above. If x is an environment variable, this edge is weighted by the probability $P(x = a | A)$, where A is the assignment corresponding to the path from the root to x .

The utility of a leaf node is the global utility $(U_1 + U_2)(A)$ of the complete assignment A associated with it. The utility of an internal decision node is given by the value of an optimal children (and it is possible to record an associated optimal decision). The utility of an internal environment node is given by the probabilistic expected utility of the values of its children nodes. The global expected utility is the utility of the root node. It can be proved [27] that such a decision tree procedure can be reduced to the computation of

$$\max_{li_J, li_P} \sum_{he_J, he_P} \max_{do} \sum_{ga, tr} ((\prod_{i \in [1, 4]} P_i) \times (\sum_{i \in [1, 2]} U_i))$$

In other words, the decision tree procedure is equivalent to a sequence of *variable eliminations* on a *combination of local functions*. Optimal decision rules can be recorded during the computation.

Different elimination sequences capture different problems or situations: if John thinks that Peter is a traitor and let him choose a door to listen in first (pessimistic attitude concerning the other agent), the sequence of elimination $\min_{li_P} \max_{li_J} \sum_{he_J, he_P} \max_{do} \sum_{ga, tr}$ is adequate (li_P is eliminated with min). If one assumes that Peter does not even tell John what he has heard (John does not observe he_P), then the sequence of elimination becomes $\min_{li_P} \max_{li_J} \sum_{he_J} \max_{do} \sum_{he_P} \sum_{ga, tr}$.

1.2 Adding feasibilities

In some cases, conditions may have to be satisfied for a decision to be feasible. Unfeasibility is *not* infinite negative utility: an adversary cannot reach the former but seeks the latter.

Example *John and Peter cannot eavesdrop to the same door and door A is locked.*

Modeling This is achieved using two local *feasibility functions*: $F_1 : li_J \neq li_P$ and $F_2 : do \neq A$. We assume that at least one decision is feasible in any situation (no dead-end). It can be represented with two normalization conditions on feasibilities: $\vee_{li_J, li_P} F_1 = t$ and $\vee_{do} F_2 = t$. A decision tree procedure to answer queries is then equivalent to compute

$$\min_{li_P} \max_{li_J} \sum_{he_J} \max_{he_P} \sum_{do} \sum_{ga, tr} ((\bigwedge_{i \in [1, 2]} F_i) \star (\prod_{i \in [1, 4]} P_i) \times (\sum_{i \in [1, 2]} U_i))$$

which uses a special operator \star for masking unfeasible decisions: if we denote the utility of unfeasible situations by \diamond , then \diamond should be an identity for elimination operators $op(e, \diamond) = e$ for any elimination operator op , and annihilator for combination operators $(\diamond \otimes e = \diamond$ for any combination operator \otimes) since the combination of an unfeasible decision with other decisions is unfeasible. Then \star is the operator that associates \diamond with unfeasible decisions: \star is defined by $f \star \alpha = \diamond$ and $t \star \alpha = \alpha$. Together, \diamond and \star exclude unfeasible situations from elimination domains.

Globally, the knowledge modeled with variables and local functions forms a *composite graphical model* defined by a DAG capturing normalization conditions on plausibilities and feasibilities (Figure 1(a))⁴, and a network of local functions (Figure 1(b)). The network involves several types of variables (decision and environment variables) and several types of local functions (local utility, plausibility and feasibility functions).

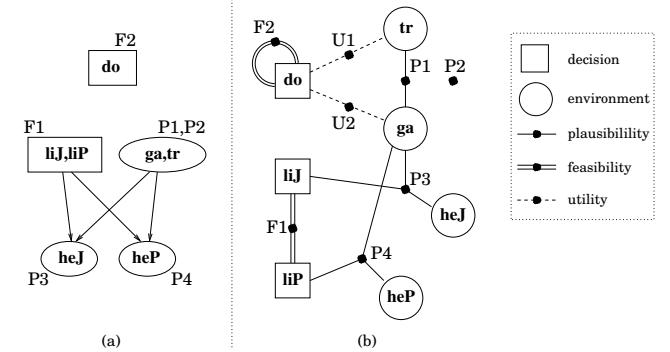


Figure 1. Composite graphical model (a) DAG capturing normalization conditions; (b) Network of local functions.

1.3 Other operators for plausibilities and utilities

The previous problem uses probabilities to model uncertainties. Under independence hypothesis, probabilities are combined with an operator $\otimes_p = \times$, and are eliminated (marginalized) with an operator $\oplus_p = +$. However, other uncertainty theories, such as possibility degrees [11] or κ -rankings [33], use other instantiations of \otimes_p/\oplus_p .

Utilities considered so far are additive (using $\otimes_u = +$). However, utilities may express preferences combined with $\otimes_u = \min$.

Probabilistic expected utility is defined as $\sum_i p_i \times u_i$: plausibilities and utilities are combined with $\otimes_{pu} = \times$ and elimination on utilities weighted by plausibilities is done with $\oplus_u = +$. Other formalisms use an expected utility defined as $\oplus_u (p_i \otimes_{pu} u_i)$ using other instantiations of \oplus_u and \otimes_{pu} :

	E_p	\oplus_p	\otimes_p	E_u	\otimes_u	\oplus_u	\otimes_{pu}
1	\mathbb{R}^+	+	\times	$\mathbb{R} \cup \{-\infty\}$	+	+	\times
2	\mathbb{R}^+	+	\times	\mathbb{R}^+	\times	+	\times
3	$[0, 1]$	\max	\min	$[0, 1]$	\min	\max	\min
4	$[0, 1]$	\max	\min	$[0, 1]$	\min	\min	$\max(1-p, u)$
5	$\mathbb{N} \cup \{\infty\}$	\min	$+$	$\mathbb{N} \cup \{\infty\}$	$+$	\min	$+$

Table 1. Operators with - 1. probabilistic expected utility - 2. probabilistic expected satisfaction - 3. optimistic and 4. pessimistic probabilistic expected utility - 5. qualitative utility with κ -rankings and with only positive utilities.

Towards a generic framework After this informal introduction, it is now possible to give a formal definition of the *Plausibility-Feasibility-Utility* (PFU) framework. This framework combines graphical models concepts (locality, conditional independence) with a flexible algebraic definition of expected utility in an *algebraic graphical model* (Section 2). We then show how queries on these networks allow to solve decision problems, with or without partial observability.

Ideally, such queries should reduce to a sequence of variable eliminations on a combination of local functions such as

$$\max_{x_1, x_2} \min_{x_3} \oplus_u \max_{x_4, x_5} \max_{x_6} ((\bigwedge_{F_i \in F} F_i) \star (\otimes_p P_i) \otimes_{pu} (\otimes_u U_i))$$

⁴ If P denotes the set of local plausibility functions associated with a node corresponding to a set of variables S , this means that $\sum_S (\prod_{P_i \in P} P_i) = 1$. If F denotes the set of local feasibility functions associated with a node corresponding to a set of variable S , this means that $\vee_S (\bigwedge_{F_i \in F} F_i) = t$.

2 An algebraic graphical model

2.1 Algebraic structure (plausibility/utility model)

Plausibility structure We define a *plausibility structure* as a triple $(E_p, \oplus_p, \otimes_p)$, where E_p is a set of plausibility degrees, equipped with a partial order \preceq_p , \oplus_p is an elimination operator on plausibilities, and \otimes_p is a combination operator on plausibilities. \oplus_p and \otimes_p satisfy some sensible axioms inspired by Friedman & Halpern's algebraic plausibility measures [14, 17]. The difference with their work is that we extend the operators \oplus_p and \otimes_p so that they become closed in E_p . This allows to define $(E_p, \oplus_p, \otimes_p)$ as a commutative semi-ring (the identity for \oplus_p is noted 0_p and the identity for \otimes_p is noted 1_p), where \oplus_p and \otimes_p are monotonic, and where 0_p (associated with impossibility) is the minimum element in E_p .

Utility structure We define a *utility structure* as a pair (E_u, \otimes_u) , where E_u is a set of utility degrees, (equipped with a partial order \preceq_u) and \otimes_u is the operator used to combine utilities. We assume that \otimes_u is monotonic and that (E_u, \otimes_u) is a commutative semi-group (\otimes_u associative and commutative, so that combined utilities do not depend on the way the combination is done).

Expected utility structure To simultaneously take into account plausibilities and utilities, we define an *expected utility structure*. This structure, inspired from Chu-Halpern's work on generalized expected utility [8] (as in algebraic MDP [26]), is a tuple $(E_p, E_u, \oplus_u, \otimes_{pu})$ where \oplus_u (operator on E_u) together with $\otimes_{pu} : E_p \times E_u \rightarrow E_u$ define the generalized expected utility formula $\oplus_u^i (p_i \otimes_{pu} u_i)$. We extend [8] in order to be able to deal with sequential decision processes. For operational reasons, \oplus_u and \otimes_{pu} are also closed. This defines $(E_p, E_u, \oplus_u, \otimes_{pu})$ as a semi-module on $(E_p, \oplus_p, \otimes_p)$, with monotonicity axioms on \oplus_u and \otimes_{pu} . Note that all the structures in Table 1 are expected utility structures.

Implicit assumptions As a set E_p of plausibility degrees and a set E_u of utility degrees are defined, plausibilities and utilities must be cardinal: purely ordinal approaches (e.g. CP-nets [4]) are not captured. As \otimes_{pu} takes values in E_u , it is implicitly assumed that plausibilities and utilities are commensurable: works as [13] are not captured either. Finally, some axioms entail that only distributional plausibilities are covered (the plausibility of a set of variable assignments is determined by the plausibilities of each covered complete assignment): belief functions [30] are not captured.

Algebraic structure for the treasure problem The plausibility structure is $(\mathbb{R}^+, +, \times)$ ($E_p = \mathbb{R}^+$ and not $[0, 1]$ in order for $+$ to be closed in E_p), the utility structure is $(\mathbb{R}, +)$, and the expected utility structure is $(\mathbb{R}^+, \mathbb{R}, +, \times)$.

2.2 A generic graphical model

Following the graphical model concepts, the environment and the decisions are now represented by environment and decision variables while the relations between these variables (plausibility, feasibility, and utility relations) are represented as local functions. Together, these define a composite and generic graphical model called a *Plausibility-Feasibility-Utility (PFU) network*.

Definition 1 A PFU network is a tuple (V, G, P, F, U) where:

- V is a set of variables, partitioned into V_D (set of decision variables) and V_E (set of environment variables).

- G is a DAG whose vertices are a partition of V (vertices of G are sets of variables called components), such that each vertex of G is composed of variables of the same nature (decision or environment). We note C_D the set of decision components and C_E the set of environment components.
- $P = \{P_1, P_2, \dots\}$ is a finite set of local plausibility functions; each P_i is associated with a component $c \in C_E$ denoted $c(P_i)$, such that the scope $S(P_i) \subset (c \cup pa_G(c))$. For any $c \in C_E$, $\oplus_p(\otimes_{p c(P_i)=c} P_i) = 1_p$ must hold.
- $F = \{F_1, F_2, \dots\}$ is a finite set of local feasibility functions; each F_i is associated with a component $c \in C_D$ denoted $c(F_i)$, such that the scope $S(F_i) \subset (c \cup pa_G(c))$. For any $c \in C_D$, we have $\vee_c (\wedge_{c(F_i)=c} F_i) = t$.
- $U = \{U_1, U_2, \dots\}$ is a finite set local utility functions.

The decomposition of global plausibility, feasibility, and utility degrees as sets of local functions is semantically justified by the notion of *conditional independence*. We provide an intuitive justification for plausibilities. Let us say that \mathcal{P}_S is a plausibility distribution on S iff $\oplus_{p_S} \mathcal{P}_S = 1_p$ (for probabilities, it simply means that a probability distribution sums up to 1). \mathcal{P}_S defines a plausibility distribution on any subset S' of S by $\mathcal{P}_{S'} = \oplus_{p_{S-S'}} \mathcal{P}_S$.

To define conditional independence, we introduce a conditioning function \otimes_p (verifying some sensible properties, see [27]) allowing to define conditional plausibility distributions by $\mathcal{P}_{S_1|S_2} = \mathcal{P}_{S_1, S_2} \otimes_p \mathcal{P}_{S_2}$ (for probabilities, \otimes_p is the division). Then, for any disjoint sets of variables S_1, S_2, S_3 , we say that S_1 is *conditionally independent* of S_2 given S_3 iff $\mathcal{P}_{S_1, S_2|S_3} = \mathcal{P}_{S_1|S_3} \otimes_p \mathcal{P}_{S_2|S_3}$.

Last, a DAG G of components is said to be *compatible* with a plausibility distribution \mathcal{P}_S iff for any component c of the DAG, c is conditionally independent of its non descendants in G given the set $pa_G(c)$ of its parents in G . Similarly to Bayesian networks [25], it is possible to prove [27] that:

Theorem 1 If G is a DAG compatible with \mathcal{P}_S , then $\mathcal{P}_S = \otimes_{p_{c \in G}} \mathcal{P}_{c|pa_G(c)}$.

Thanks to Theorem 1⁵, a DAG compatible with a global plausibility distribution enables to factorize it as a combination of local functions $L_{c, pa_G(c)}$ ($L_{c, pa_G(c)} = \mathcal{P}_{c|pa_G(c)}$) such that $\oplus_{p_c} L_{c, pa_G(c)} = 1_p$ for any component c of G . Each $L_{c, pa_G(c)}$ may be further decomposed to give local plausibility functions P_i associated with c (denoted $c(P_i) = c$) thus verifying $\oplus_{p_c} (\otimes_{p_{c(P_i)=c}} P_i) = 1_p$. This two-step factorization justifies the DAG in a PFU network and the normalization conditions it encodes.

Similar results can be established for feasibilities. As for utilities, PFU networks implicitly assume that a global utility degree \mathcal{U}_V on all variables can be decomposed as a set of local utility functions U_i . One may assume that this decomposition is directly obtained, as it is done with CSP [22] or is justified by a notion of conditional independence in the case $\otimes_u = +$ as in [1].

Example The PFU network of our example appears in Figure 1.

Subsumption results CSP [22] can be modeled as $(V, G, \emptyset, \emptyset, U)$ where V contains the CSP variables, the DAG G is reduced to one decision component equal to V and U is the set of constraints (by normalization, feasibilities cannot represent inconsistent CSP).

⁵ And with some technical steps induced by the fact that we do not work on plausibility distributions on V , but on plausibility distributions on V_E for any assignment of V_D .

It is also possible to capture valued [2], quantified [3], mixed [12], stochastic [34] CSP, or SAT, QBF, and extended stochastic SAT [21].

Bayesian networks [25] are captured by a PFU network $(V, G, P, \emptyset, \emptyset)$, where P contains the original probability tables. Chain graphs [15], Markov random fields [7] are also subsumed.

Finite-horizon probabilistic MDP [28, 23] are captured by (V, G, P, \emptyset, U) where V_D and V_E are the set of decisions d_t and states $s - t$ respectively (one for each time-step t), G is a DAG looking like the unrolled MDP, P contains the probability distributions $P_{s_{t+1} | s_t, d_t}$, and U contains the additive rewards R_{s_t, d_t} . It is also possible to model finite-horizon probabilistic MDP [29], MDP based on κ -rankings, partially observable (PO) MDP, factored or not [5, 6].

Influence diagrams [18], including a probabilistic variant, can be represented as a tuple (V, G, P, \emptyset, U) . For valuation networks [32], a set F of local feasibility functions is added.

3 Reasoning about PFU networks via queries

In this section, we assume that a *sequence of decisions* must be performed, and that the order in which decisions are made and the environment is observed is known. We also make a *no-forgetting* assumption, that is, when making a decision, an agent is aware of all previous decisions and observations. Finally, the order on utility degrees is assumed to be total.

Under such assumptions, we want to express sequential decision making problems on PFU networks, taking into account possible partial observability and cooperative or antagonist agents in the environment. To capture such problems, we use a sequence Sov of operator-variable(s) pairs that captures:

- *possible unobservabilities*: the order in which decisions are made and environment variables observed is specified by Sov . If the value of he_P is known when John chooses a door to open, then Sov contains $\dots(\oplus_u, he_P)\dots(\max, do)\dots$. Otherwise, a sequence like $\dots(\max, do)\dots(\oplus_u, he_P)\dots$ is used;
- *optimistic/pessimistic attitude* concerning the decision makers: if Peter acts cooperatively (we are optimistic about Peter's decision) then (\max, li_P) appears in Sov . If instead Peter is considered as an antagonist agent then (\min, li_P) is used.

Example For the last query in §1.1, Sov equals $(\min, \{li_P\}).(\max, \{li_J\}).(+, \{he_J\}).(\max, \{do\}).(+, \{he_P\}).(+, \{ga, tr\})$.

Definition 2 A query Q on a PFU network is a pair (\mathcal{N}, Sov) where (1) \mathcal{N} is a PFU network; (2) Sov is a sequence of operator-variable(s) pairs such that the operators are \min , \max or \oplus_u , and such that each variable appears at most once in Sov .

Correct queries Not all queries are meaningful. The main condition for a query to be *correct* is that it must not contain a pair x, y of variables of different nature such that x belongs to an ascendant component of y in the DAG of the PFU network and x appears after y in Sov . This would mean that x is assigned after y , breaking causality. For example, the pair $\dots(+, he_J)\dots(\max, li_J)\dots$ is not correct since John cannot hear something at the door he has chosen to eavesdrop before this choice is done.

3.1 Answering queries

Answering a query Q consists in computing the expected utility associated with the situation modeled by the sequence Sov and the network \mathcal{N} of the query Q .

Decision tree approach A first approach to answer queries in the general case (not only for probabilistic expected utility) uses decision trees. In this case, variables are considered as they appear in Sov , and edges in the tree are weighted by conditional plausibilities of the form $\mathcal{P}(x = a | A)$ for the internal nodes associated with environment variables, and by conditional feasibilities of the form $\mathcal{F}(x = a | A)$ for internal nodes associated with decision variables.

Then the expected utility of a query (and associated optimal decision rules specifying which decision to take given the previous observed variables) can be defined with a decision tree procedure similar to the procedure described in 1.1 (the utility of a leaf node is given by the combination of the local utilities U_i , the utility of an internal environment node is given by the expected utility of its children, and the utility of an internal decision node is given by the optimal utility of its feasible children nodes).

A more operational approach The advantage of the decision tree procedure is that it has clear semantic foundations. But besides the possibly exponential size tree, its drawback is that each internal node of the decision tree may require the computation of $\mathcal{P}(x = a | A)$ or $\mathcal{F}(x = a | A)$ which are not usually directly available in the network \mathcal{N} and which may require exponential time to compute. Fortunately, it is possible to show [27] that the decision tree procedure (called the semantic answer to Q) is equivalent to a direct algebraic approach (called the operational answer to Q) which requires only the local functions available in the original PFU network.

Theorem 2 Answering a query with a decision tree is equivalent to compute $Ans(Q) = Sov((\bigwedge_{F_i \in F} F_i) * (\bigotimes_{P_i \in P} P_i) \otimes_{pu} (\bigotimes_{U_i \in U} U_i))$

Furthermore, the optimal decision rules obtained are the same as in the decision tree approach.

3.2 Subsumption of classical queries

Most usual queries on existing formalisms can be reduced to PFU queries: finding a solution for a SAT problem, a CSP [22] or a valued CSP [2] corresponds to a sequence $Sov = (\max, V)$. For QBF or quantified CSP [3], Sov alternates \min (for universal quantification) and \max (for existential quantification). With mixed or probabilistic CSP [12], Sov looks like $(\oplus_u, V_E).(\max, V_D)$ if a conditional decision is sought and $(\max, V_D).(\oplus_u, V_E)$ if an unconditional decision is sought. The situation is similar with conformant or probabilistic planning [16].

Queries on Bayesian networks [25] look like $(+, S)$ to compute a probability distribution on $V - S$, (\max, V) to solve a Most Probable Explanation problem, and $(\max, V_D).(+, V_E)$ to solve Maximum A Posteriori problems.

With stochastic CSP [34] or influence diagrams [18], the sequence alternates $+$ on environment variables and \max on decision variables. With finite-horizon MDP, Sov looks like $(\max, d_1).(\oplus_u, s_2)\dots(\max, d_T).(\oplus_u, s_T)$. With finite-horizon POMDP, observations o_t are added (one for each time-step t), and $Sov = (\max, d_1).(\oplus_u, o_2)\dots(\max, d_T).(\oplus_u, o_T).(\oplus_u, \{s_1, \dots, s_T\})$: this captures the fact that states remains unobserved for POMDP.

4 Gains and costs

A better understanding As it subsumes many queries on existing graphical models, the PFU framework enables to better understand the similarities and differences between the subsumed formalisms.

It defines a common basis for people of different communities to communicate.

Increased expressiveness The PFU framework offers several variabilities: (1) variability of the algebraic structure, which captures probabilistic expected utility, probabilistic expected satisfaction, possibilistic pessimistic utility, possibilistic optimistic utility or qualitative utility with κ -rankings; (2) variability of the network which exploits oriented and non-oriented independences as well as normalization conditions; (3) variability of the queries which can capture state (un)observability and cooperative/antagonist attitudes.

It is therefore more expressive than each of the frameworks it subsumes, and it also covers yet unpublished formalisms (such as possibilistic influence diagrams, or stochastic CSP extended to cope with the fact that decisions may influence the environment).

Generic algorithms Computing the answer to a correct query is obviously PSPACE-hard since PFU queries capture QBF. It is easy to define, from Theorem 2, a polynomial space tree search algorithm which computes the answer to a correct query. Similarly, it is possible to define a generic variable elimination algorithm to compute $Ans(Q)$ [10]. The PFU algebraic framework is an opportunity to identify sufficient or necessary conditions for existing algorithms to be applicable [24], or to define new techniques from which each subsumed formalism could benefit. Bounding and local consistencies [22, 9, 20] could be integrated to speed up the resolution.

As a result, the PFU framework can be seen as an opportunity to integrate in a generic framework techniques developed in different subsumed formalisms, and thus to allow for cross-fertilization.

5 Conclusion

In this paper⁶, a generic algebraic framework for sequential decision making has been defined. It combines an algebraic structure that specifies how to combine and synthesize information together with a graphical model specifying local plausibility, feasibility, and utility functions. Queries can capture possible (un)observabilities or antagonist agents.

The generalized expected utility associated with a query can be computed by a sequence of variable eliminations on a combination of local functions. Compared to *valuation algebras* [31, 19], a related generic framework, the PFU framework uses several combination (\wedge , \star , \otimes_p , \otimes_{pu} , \otimes_u) and elimination operators (min, max, \oplus_u). Moreover, the semantic justifications of the definition of PFU networks, which lie in the notion of conditional independence, allow to include a DAG capturing normalization conditions in the network definition.

The obtained framework not only subsumes many queries on existing formalisms, but it also enables to define yet unpublished formalisms. From an algorithmic point of view, generic schemes that integrate techniques used in subsumed formalisms can be developed.

References

- [1] F. Bacchus and A. Grove, ‘Graphical Models for Preference and Utility’, in *Proc. of UAI*, (1995).
- [2] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier, ‘Semiring-Based CSPs and Valued CSPs: Frameworks, Properties and Comparison’, *Constraints*, **4**(3), 199–240, (1999).
- [3] L. Bordeaux and E. Monfroy, ‘Beyond NP: Arc-consistency for Quantified Constraints’, in *Proc. of the 8th International CP conference*, Ithaca, New York, USA, (2002).
- [4] C. Boutilier, R. Brafman, H. Hoos, and D. Poole, ‘Reasoning With Conditional Ceteris Paribus Preference Statements’, in *Proc. of the 15th UAI Conference*, Stockholm, Sweden, (1999).
- [5] C. Boutilier, R. Dearden, and M. Goldszmidt, ‘Stochastic Dynamic Programming with Factored Representations’, *Artificial Intelligence*, **121**(1-2), 49–107, (2000).
- [6] Craig Boutilier, Thomas Dean, and Steve Hanks, ‘Decision-theoretic planning: Structural assumptions and computational leverage’, *Journal of Artificial Intelligence Research*, **11**, 1–94, (1999).
- [7] R. Chellappa and A. Jain. *Markov random fields: Theory and applications*. Academic Press, 1993.
- [8] F.C. Chu and J.Y. Halpern, ‘Great expectations. part i: On the customizability of generalized expected utility’, in *Proc. of IJCAI*, pp. 291–296, Acapulco, Mexico, (2003).
- [9] M. Cooper and T. Schiex, ‘Arc Consistency for Soft Constraints’, *Artificial Intelligence*, **154**(1-2), 199–227, (2004).
- [10] R. Dechter, ‘Bucket Elimination: A Unifying Framework for Reasoning’, *Artificial Intelligence*, **113**(1-2), 41–85, (1999).
- [11] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, 1988.
- [12] H. Fargier, J. Lang, and T. Schiex, ‘Mixed Constraint Satisfaction: a Framework for Decision Problems under Incomplete Knowledge’, in *Proc. of the 13th AAAI*, pp. 175–180, Portland, USA, (1996).
- [13] H. Fargier and P. Perny, ‘Qualitative Models for Decision Under Uncertainty without the Commensurability Assumption’, in *Proc. of UAI*, (1999).
- [14] Friedman N., Halpern J., ‘Plausibility Measures : A User’s Guide’, in *Proc. of UAI*, Montreal, Canada, p. 175–184, (1995).
- [15] M. Frydenberg, ‘The Chain Graph Markov Property’, *Scandinavian Journal of Statistics*, **17**, 333–353, (1990).
- [16] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004.
- [17] J.Y. Halpern, ‘Conditional plausibility measures and Bayesian networks’, *JAIR*, **14**, 359–389, (2001).
- [18] R. Howard and J. Matheson, ‘Influence Diagrams’, in *Readings on the Principles and Applications of Decision Analysis*, 721–762, Strategic Decisions Group, Menlo Park, CA, USA, (1984).
- [19] J. Kolhas, *Information Algebras: Generic Structures for Inference*, Springer, 2003.
- [20] J. Larrosa and T. Schiex., ‘In the quest of the best form of local consistency for weighted CSP’, in *Proc. of IJCAI*, Acapulco, Mexico, (2003).
- [21] M. Littman, S. Majercik, and T. Pitassi, ‘Stochastic Boolean Satisfiability’, *Journal of Automated Reasoning*, **27**(3), 251–296, (2001).
- [22] A. Mackworth, ‘Consistency in Networks of Relations’, *Artificial Intelligence*, **8**(1), 99–118, (1977).
- [23] G. Monahan, ‘A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms’, *Management Science*, **28**(1), 1–16, (1982).
- [24] P. Ndiliikilesha, ‘Potential Influence Diagrams’, *International Journal of Approximated Reasoning*, **3**, 251–285, (1994).
- [25] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.
- [26] P. Perny, O. Spanjaard, and P. Weng, ‘Algebraic Markov Decision Processes’, in *19th IJCAI Proc.*, Edinburgh, Scotland, (2005).
- [27] C. Pralet, G. Verfaillie, and T. Schiex. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities, and Utilities. LAAS-CNRS Report, <http://www.laas.fr/~cprale/praleverfschiex.ps>, 2005.
- [28] M. Puterman, *Markov Decision Processes, Discrete Stochastic Dynamic Programming*, John Wiley & Sons, 1994.
- [29] R. Sabbadin, ‘A Possibilistic Model for Qualitative Sequential Decision Problems under Uncertainty in Partially Observable Environments’, in *Proc. of the 15th UAI Conference*, Stockholm, Sweden, (1999).
- [30] G. Shafer, *A mathematical theory of evidence*, Princeton University Press, 1976.
- [31] P. Shenoy, ‘Valuation-based Systems for Discrete Optimization’, *Uncertainty in Artificial Intelligence*, **6**, 385–400, (1991).
- [32] P.P. Shenoy, ‘Valuation network representation and solution of asymmetric decision problems’, *European Journal of Operational Research*, **121**, 579–608, (2000).
- [33] W. Spohn, ‘A general non-probabilistic theory of inductive reasoning’, in *Uncertainty in Artificial Intelligence 4*, 149–158, North-Holland, Amsterdam, (1990).
- [34] T. Walsh, ‘Stochastic Constraint Programming’, in *Proc. of the 15th ECAI Conference*, Lyon, France, (2002).

⁶ This work was partially conducted within the EU IP COGNIRON (“The Cognitive Companion”) funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

Using Occlusion Calculi to Interpret Digital Images

David Randell¹ and Mark Witkowski¹

Abstract. This paper reports on an investigation using occlusion calculi to interpret digital images. Using a minimal set of digital, region-relation detectors, and assuming a continuous interpretation of physical space, we show how existing calculi can be augmented and embedded in the *Event Calculus* to interpolate and recover a larger set of occlusion relations than are otherwise available at the basic detector level.¹

1 INTRODUCTION

Abstracting and reasoning about high-level symbolic spatial information about extended bodies (or regions) has long been a motivating ontological assumption used in *Qualitative Spatial Reasoning (QSR)* research. That several suitably expressive *QSR* logics have subsequently turned out to have interesting and useful computational properties has added to this general degree of confidence in their use [2,3]. *QSR* applications cover such diverse areas as querying spatial databases linked to *GIS*, the encoding of vague concepts, and in processing visual information using real-world images [3,10]. In terms of practical applications however, few have applied themselves to the task of deploying these logics on simulated or real-world machine-vision systems.

Given the relative absence of published work along these latter lines, plus the fact that *QSR* has now been an active research topic in AI for at least 17 years, it is perhaps time to enquire why, and despite the large number of spatial logics *QSR* has spawned, *QSR* has not made more headway here than it has. To this end, we have honed in on a subset of *QSR* logics known as *occlusion calculi* [5, 8, 10, 11] and in particular have singled out Galton's *Lines Of Sight (LOS-14)* calculus [5] with which to interpret 3D convex bodies in terms of their projected images. We wish to examine to what extent occlusion calculi can be used to interpret idealised, real-world two-dimensional digital images returned by a camera.

If we assume (as we do) that our regions are derived directly as the segmented images of physical bodies, then spatial occlusion events becomes a significant issue as several ontological assumptions of existing *QSR* logics (e.g. that the extent of the regions in question is known *a priori*, and that the embedding space is continuous) no longer hold. As we show, these and other assumptions have significant ramifications for digital geometry [7] and *QSR* logics in general.

In order to strengthen and simplify our example, we (i) modify the ontological base on which the original *LOS-14* calculus was built to handle a *discrete* (digitised) rather than a continuous model of space, (ii) embed the logic in an abductive inferential framework using the *Event Calculus* and exploit Conceptual Neighbourhood Diagrams (CNDs) in order to interpolate (by abduction) spatial relations not directly encoded in our set of region-relation detectors, (iii) simplify the segmentation process by individuating regions as aggregates of pixels falling within a specific colour-hue range, and (iv) assume a computational model of region identity, to track regions over time (e.g. [12]).

The rest of this paper is structured as follows. In section 2 we introduce occlusion calculi. Three important limiting assumptions are challenged, (i) that the extent of regions is known *a priori*, (ii) that embedding space is continuous, and (iii) that region boundary information can be reliably extracted from real images. In section 3 we introduce a discrete mereotopological theory. This is used to define the logical properties of regions and relations extracted from segmented digitised images, and specify the logical properties of a small set of region, and region-relation detectors. In section 4 the detectors are mapped to the relation set of *LOS-14*. Section 5 presents an overview of a practical implementation of the detectors. In section 6, the *Event Calculus* is used to abductively infer occlusion relations from an assumed temporal succession of detector states extracted from a sequence of images. Concluding remarks are made in section 7.

2 OCCLUSION CALCULI

Region-based Occlusion Calculi for vision applications originated with Galton's *Lines of Sight* paper (*LOS-14*) [5]; this was followed by *The Region Occlusion Calculus (ROC-20)* [10], [11], and Kohler's *Occlusion Calculus* [8]. These calculi spring from the general development of *QSR* formalisms, calculi and algebras that have been developed over the years for modelling sub-theories of space. In contrast to other *QSR* spatial formalisms, occlusion calculi adopt a distinctive *viewpoint-centred* model of space where visual spatial relations on bodies (or regions) are mapped to corresponding spatial relations on their images.

At least two of these logics (*LOS-14* and *ROC-20*) assume a continuous embedding model of space, and each factors out a *Jointly Exhaustive and Pairwise Disjoint (JEPD)* set of relations. Constraints on the interpretation of the regions, primitive functions and relations used, mean the degree of theory overlap between them varies. For example, Galton's *LOS-14*, defined on discrete convex bodies, identifies 14 *JEPD* relations (see Figure 1), whereas *ROC-20* relaxing this convexity constraint identifies 20. All the relations defined in *LOS-14* have analogues in *ROC-20*, and both are built on the set of relations identified in the spatial logic *RCC-8* [9]. In contrast, Kohler's *Occlusion Calculus* builds on the weaker set of relations defined in the spatial logic *RCC-5* [2].

As is common with many *QSR* theories, each *JEPD* relation set is re-worked into a conceptual neighbourhood diagram (*CND*). These diagrams encode the continuous transitions allowed between pairs of spatial relations [4]. For example, in *LOS-14*, a direct transition exists between *C/2* (clears), and *JC/2* (just clears) but not between *C/2* and *PH/2* (partially hides); where here and elsewhere, the *R/n* notation used, identifies an n-ary relation. The same *CND* can also be used to justify interpolated relations, as in the case where *JC/2* is inferred given a detected transition between *C/2* and *PH/2* – see Figure 1.

¹ Dept. of Computing, Imperial College London, UK.
 email:{d.randell,m.witkowski}@imperial.ac.uk

2.1 Occlusion Calculi: some assumptions

Given that all the occlusion calculi described here are to a large extent, theoretically motivated, their ontological assumptions need to be carefully examined given we seek to ground the logical primitives in real image data.

The first assumption noted is that the extent of the bodies is assumed known *a priori*, or is at least directly inferable from other information present. However, in the presence of opacity, the mode and extent of visual overlap between bodies is more often than not indeterminate. For example, consider the *CND* of *LOS-14* depicted in Figure 1. Of the 14 possible occlusion relations, 12 rely on part of one body being hidden by another, while only 8 have both bodies directly visible. On the assumption that the physical bodies giving rise to image regions are opaque, and therefore that their hidden extent cannot be determined from any given viewpoint, only three distinct relations of this type can be reliably distinguished. This is discussed in detail later.

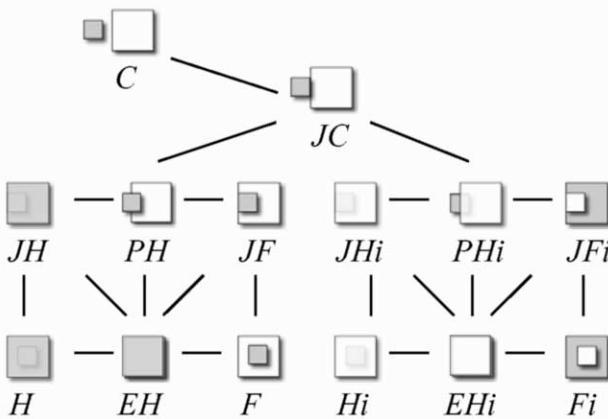


Figure 1: The conceptual neighbourhood diagram LOS-14

The second assumption is that all the existing occlusion calculi assume a *continuous* model of space. As is now well known, the spatial logic *RCC8* used in [5,7,9] disallows *atomic* regions; otherwise the whole calculus is rendered inconsistent [9]. Yet clearly, the space modelled by pixel-arrays is discrete, and if individual pixels here cannot be treated as atomic regions, how then can digital images provide a model for these calculi? The continuous vs. discrete issue for *QSR* formalisms is also discussed in [6], [13] where discrete variants of the *RCC8* calculus are motivated and developed. Also, and related to this point, is how we interpret the primitives of these logics. Take for example the relation *C/2* used in *LOS-14* (see Figure 1), or its analogue relation *NonOccludesDC/2* used in *ROC-20*. Whether we choose to interpret these relations in terms of lines of sight (as in *LOS-14*) or as a pencil of half-lines extending from some viewpoint point v , to infinity (as in *ROC-20*), a fundamental difference arises when using the projective plane defined on the real numbers compared with a viewer-centred discrete interpretation of space. In the first case, the spatial relation $C(A,B)$ is always true whatever the image scale assumed. But in the latter case, whether or not A and B are visually separated depends upon other parameters of the physical system being modelled, such as the image scale or the resolution of the imaging system. This is, of course, an example of the *granularity* issue that is well known in AI research.

The third and last assumption highlighted here is that region-boundary information can be reliably extracted from digitised images. The emphasis on boundaries is a direct consequence of adopting mereotopological logics, for it is precisely the mode of

contact between the visual boundaries of regions that determines what spatial relation is true. However, the reliable interpretation of boundary information from real-world digital images is problematic and detectors based on this information tend to be brittle and often unreliable in use.

These assumptions are of course by no means exhaustive. For example, sensor noise, fast sampling rates, the effects of variable lighting conditions, of shadows and specular reflections all affect the segmentation of the raw digital data returned by any current physical imaging system. However, the main assumptions stemming from the ontology and models used in occlusion calculi have been identified. It is specifically these assumptions that we now address in a *dynamic QSR* framework.

3 DISCRETE MEREOPOLOGY

We use Galton's *two-sorted mereotopological logic* [6] as a *specification language* to define the logical properties of our region and region-relation detectors defined on aggregates of pixels as regions. Two sorts are declared: *pixels* (*cells* in Galton's case) and *regions* as aggregates of pixels.

In [6], definitions of the classical mereological relations and the Boolean composition of regions are given and interpreted set-theoretically. A null region \emptyset is declared, that contains no pixels, and the universal region U , that contains every pixel in the embedding space. Containment of a pixel x in a region X is modelled as set membership, i.e. as $x \in X$; and adjacency of two pixels is captured by the reflexive, symmetric relation $A(x,y)$, meaning pixel x is adjacent to or equal to pixel y . We use a square-based pixel array, and adjacency here is interpreted as 8-adjacency, meaning every non-boundary pixel of the array is surround by 8 neighbours forming a combined 3x3 pixel matrix.

The basic relation-based definitions proceed as follows. Inclusion between regions is defined as: $X \subseteq Y \equiv \forall x \in X \rightarrow x \in Y$, and the (non-null) part/whole relation as: $P(X,Y) \equiv X \subseteq Y \wedge X \neq \emptyset$. Overlap is defined as: $O(X,Y) \equiv X \cap Y \neq \emptyset$. For the weaker relation of connection (or contact) between regions, the adjacency relation now appears: $C(X,Y) \equiv \exists x \exists y [x \in X \wedge y \in Y \wedge A(x,y)]$. This enables the external connection relation: $EC(X,Y)$ and the remaining discrete analogue counterparts of the *RCC8* relations and other related concepts (For example, Galton defines neighbourhood, boundary, interior, closure and covers metric spaces, change and continuity) to be defined. To avoid confusion with the *RCC8* relations these will be referred to as *RCC8D* [6]. We will use only three of these *RCC8D* relations: DisConnected, $DC(X,Y)$; Externally Connected, $EC(X,Y)$; and Non Tangential Proper Part, $NTPP(X,Y)$, mirroring the detectors. Taking $U-X$ as the region difference between U and X , these are defined as follows: $DC(X,Y) \equiv \neg C(X,Y)$, $EC(X,Y) \equiv C(X,Y) \wedge \neg O(X,Y)$, and $NTPP(X,Y) \equiv DC(X,U-Y)$.

While *CNDs* exist for discrete mereotopological logics (e.g. *RCC8D*), the effect of the digital sampling serves to significantly increase the number of possible distinct relation-relation transitions when compared with their continuous counterparts. Here, we assume an underlying continuous model of physical space while sampling the relations into a discrete representation. This has the advantage of (i) allowing physically realisable interpretations of 'anomalous' transitions close to the resolution limit of the detectors, and (ii) exploiting much stronger constraints on possible transitions than directly supported by the discrete model of space.

3.1 Extensions

A finite disjoint set of individual region-membership predicates: R_1, \dots, R_n , is defined on aggregates of pixels. The exact nature of the membership property R_i need not concern us here; but will typically range over some physical property predicated on bodies, such as colour as assumed in our worked example. As uniform patches of colour are used as the basis of both object (or body) and region-identity, and is used by our region segmentation algorithm to label individual regions, we can state the equivalence: $R(X) \equiv \forall x/x \in X \rightarrow R(\{x\})$. Another property, that of a maximally-connected region, $\text{Max}(X)$, is also adopted; in which aggregates of adjacent pixels satisfying property R_i are clustered into connected subsets. This ensures a minimal number of explicitly labelled regions at the detector level: $\text{Max}(X) \equiv V_{i=1}^n [R_i(X) \wedge \forall x \forall y/x \in X \wedge y \notin X \wedge A(x,y)] \rightarrow \neg R_i(\{y\})$. While not developed here, individual regions can then be classified into connected and disconnected (i.e. scattered) regions, see [6] for details. Finally, an additional property of *well-formed region* (wfr) is defined on aggregated sets of pixels as satisfying the definition: $\text{Region}(X) \equiv V_{i=1}^n [R_i(X) \wedge \forall x/x \in X \rightarrow \exists y \exists z \exists u [y \in X \wedge z \in X \wedge u \in X \wedge x \neq y \wedge x \neq z \wedge x \neq u \wedge y \neq z \wedge u \neq z \wedge A(x,y) \wedge A(x,z) \wedge A(u,y)]]$. A well-formed region defines a set of pixels X , satisfying some property R , where every pixel element of X has at least 3 distinct immediate neighbours of that set. Regions not satisfying this condition are passed to our algorithm for pre-processing prior to segmentation and labelling of the regions.

4 REGION CONTACT DETECTORS

The simplest detectors report whether or not a labelled region X (from viewpoint v) is detected: $\text{visible}(X,v) = \text{true}$, iff from v , pixel-region X is detected in the pixel-array, otherwise false. This allows us to infer when, for example, $\text{visible}(a,v) = \text{false}$, that a exists but is hidden from v . These relations (with their reified regions) re-appear as the primitive relation: $\text{Visible}(x,v)$ in the object language.

The pixel-aggregate detector definitions for DC^* , EC^* and $NTPP^*$ are now as follows (see Figure 2 for examples):

$$\begin{aligned} DC^*(X,Y) &\equiv V_{i=1,j=1}^n [R_i(X) \wedge R_j(Y)] \wedge \\ &\quad \neg \exists x \exists y [x \in X \wedge y \in Y \wedge A(x,y)] \\ EC^*(X,Y) &\equiv V_{i=1,j=1}^n [R_i(X) \wedge R_j(Y)] \wedge \\ &\quad \exists x \exists y [x \in X \wedge y \in Y \wedge A(x,y)] \wedge \neg NTPP^*(X,Y) \\ NTPP^*(X,Y) &\equiv V_{i=1,j=1}^n [R_i(X) \wedge R_j(Y)] \wedge \\ &\quad [NTPP(X,Y) \wedge Y' = X + Y] \end{aligned}$$

Each of these relations are mapped to three implemented detectors, dc^* , ec^* and $ntpp^*$, with $ntpp^*$ as the inverse of $ntpp$. Note that the $ntpp^*$ detector does not of itself embody the transitive property of the (*RCC8* and *RCC8D*) relation $NTPP/2$, as the detector operates locally. Thus, if $ntpp^*(A,B)$ and $ntpp^*(B,C)$, then $dc^*(A,C)$ will be initially reported. However, as the domain model restricts bodies to convex regions, we infer $ntpp^*(A,C)$ via its mapping to $NTPP/2$. The justification for labelling embedded nested regions as $NTPP^*$ (as shown in Figure 2) follows from the (*RCC8* and *RCC8D*) theorem: $\forall xyz [NTPP(\text{sum}(x,y),z) \rightarrow [NTPP(x,z) \wedge NTPP(y,z)]]$, i.e. every part of a non-tangential part of region forms a non-tangential relationship to the whole.

In terms of the *LOS-14* occlusion relations mapping to the relation detectors, only one is clearly unambiguous, namely C , which maps to DC^* ; while the others: JC, PH, JF, JFi map to EC^* ; and F , which maps to $NTPP^*$, are either ambiguous or

reintroduce some other assumption into the interpretation. In the case of the latter, for example, a one-one mapping is only possible if we make (as we do) the additional convexity assumption. Having now reduced the 14 occlusion relations down to a detector that can only distinguish three classes of relations, how then can we recover the missing ones?

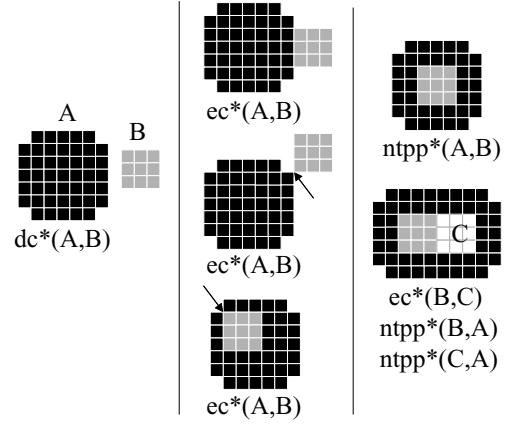


Figure 2: Discrete digitised regions

4.1 Mapping between Bodies and Regions

To illustrate the interplay between bodies and digitised regions we now embed the relations: DC^* , EC^* and $NTPP^*$ (and the inverse relation $NTPP^*$) into *LOS-14*. An explicit viewpoint variable, v , and an image function $\text{image}(x,v)$, that maps body x , and viewpoint v to its projected image is added, and each dyadic *LOS-14* relation, is re-worked into a ternary relation – see [10] for further details.

The first set of axioms map the occlusion relations to their *RCC8D* spatial relations on images; then following the group of definitions, another group of axioms map between selected *LOS-14* relations and the primitive region-relation detectors. For brevity we omit from the antecedents of these axioms conjuncts that encode the visibility or otherwise of bodies given particular occlusion relationships. However, these can be inferred from the model used to illustrate the CND in Figure 1.

$$\begin{aligned} \forall x \forall y \forall v [C(x,y,v) &\rightarrow DC(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [JC(x,y,v) &\rightarrow EC(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [PH(x,y,v) &\rightarrow PO(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [JF(x,y,v) &\rightarrow TPP(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [F(x,y,v) &\rightarrow NTPP(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [JH(x,y,v) &\rightarrow TPPi(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [H(x,y,v) &\rightarrow NTPPi(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [EH(x,y,v) &\rightarrow EQ(\text{image}(x,v), \text{image}(y,v))] \\ PHi(x,y,v) &\equiv \text{def. } PH(y,x,v), JHi(x,y,v) \equiv \text{def. } JH(y,x,v) \\ JFi(x,y,v) &\equiv \text{def. } JF(y,x,v), Hi(x,y,v) \equiv \text{def. } H(y,x,v) \\ Fi(x,y,v) &\equiv \text{def. } F(y,x,v), EH(x,y,v) \equiv \text{def. } EH(y,x,v) \\ \forall x \forall y \forall v [C(x,y,v) &\rightarrow DC^*(\text{image}(x,v), \text{image}(y,v))] \\ \forall x \forall y \forall v [\Phi(x,y,v) &\rightarrow EC^*(\text{image}(x,v), \text{image}(y,v))] \\ \text{where: } \Phi &\in \{JC, PH, JF, JFi, PHi\} \\ \forall x \forall y \forall v [F(x,y,v) &\rightarrow NTPP^*(\text{image}(x,v), \text{image}(y,v))] \end{aligned}$$

5 THE DETECTOR ALGORITHM

This section summarises the main points of the algorithm (and program) used to implement the detectors and transform input

bitmap images to regions and extract the QSR relations between them. The (C++) program accepts as input bitmap images (of size x by y) and, in step 1, assigns a region value in the range 0 to $\text{maxr}-1$ (max regions) for each unique colour value encountered (the array r). Test images may be prepared using the block fill tool in any standard bitmap manipulation program. The program also allows camera images to be grabbed directly using a Videre Mega-D firewire camera or imported via bitmap images. Prior to use, real images are subjected to a hue separation process to extract distinct regions. The resulting region based image is subjected to a region-conditioning process that implements the notion of well-formed region (wfr) introduced earlier in section 3.1. Every pixel with fewer than three like neighbours is converted to the identity of the majority of adjacent pixels (or arbitrarily where the count is equal). The process is applied iteratively until the segmentation results in a set of labelled regions X_1, \dots, X_n that satisfy $\text{Region}(X_1) \& \dots \& \text{Region}(X_n)$. This sequence of activities is indicated below by `get_regions()`. The process has the additional effect of overcoming the effects of region boundary pixel aliasing inherent in digital imaging, and of removing “speckle” noise pixel groups. The image regions are considered to be embedded in a background area, which is not taken as a region.

In step 2 the `relations` array, which records the detector relations between all combinations of regions, is initialised to `dc*` (`DCSTAR`). The image sized array `mask` is used to track those pixels and regions that have been processed. It is initialised to discount all background pixels.

In step 3 each region value is compared to its neighbour (here, `neighbour()` returns the coordinates of an adjacent pixel) and processes each region as it encounters each new region edge. `TraceOutline()` recursively traces around the outside edge of the new region building a mask (`omask`) of those connected edge pixels. Note that here `all_neighbours()` successively returns the coordinates of all eight connected pixels. Next, `FillRegion()` applies a recursive flood-fill to the current region, noting the edge pixels encountered (`IsEdge()`). Any edge pixel (that also belongs to a region) that has previously been tagged in `omask` as an outside region indicates that the current and touching regions form a symmetric `ec*` relationship (`SetECRegion()`), and that any other connected region must therefore form an (asymmetric) internal `ntpp*/ntppi*` relationship (`SetNTPPRegion()`). Note that under this arrangement evidence of an `ntpp*` relation effectively takes precedence over that for an `ec*` one. This process is repeated until all regions have been processed. Finally the `relations` array is displayed and reformatted for further processing (Figure 4).

```

process(raw_image)
{ // Step 1: convert image to regions
  proc_image = get_regions(raw_image);
  forevery(x, y)
    r(x,y) := ToRegionValue(proc_image[x,y]);
  // Step 2: initialise relations[] to dc*
  forevery(i < maxr, j < maxr)
    relations[i,j] := DCSTAR;
  initialise(mask);
  // Step 3: process all regions
  forevery(x, y)
    { p = r[x,y]
      if(p != r[neighbour(p)] && !mask[neighbour(p)])
        { clear(omask);
          cval := r[neighbour(p)];
          TraceOutline(cval,x,y);
          FillRegion(cval,x,y); } }
  Display_regions(r); }
TraceOutline(cval,x,y)

```

```

{ if(r[x,y] == cval) return;
  if(omask[x,y]) return;
  if(IsOuterEdge(cval,x,y)) return;
  omask(x,y) := 1;
  forevery({x',y'} := all_neighbours(x,y))
    TraceOutline(cval,x',y'); }

FillRegion(cval,x,y)
{ if(mask[x,y]) return;
  if(r[x,y] != cval) return;
  IsEdge(cval,x,y);
  mask[x,y] := 1;
  forevery({x',y'} := all_neighbours(x,y))
    FillRegion(cval,x',y'); }

bool IsOuterEdge(cval,x,y)
{ result := false;
  if(r[x,y] == cval) return(result);
  forevery({x',y'} := all_neighbours(x,y))
    { if(r[x',y'] == cval)
      { omask[x',y'] := 1; result := true; } }
  return(result); }

IsEdge(cval,x,y)
{ forevery({x',y'} := all_neighbours(x,y))
  { if(cval != r(x',y'))
    if(omask[x',y'])
      SetECRegion(x,y,x',y');
    else
      SetNTPPRegion(x,y,x',y'); }

SetECRegion(x1,y1,x2,y2)
{ if(relations[r[x1,y1],r[x2,y2]] == DCSTAR)
  { relations[r[x1,y1],r[x2,y2]] = ECSTAR;
    relations[r[x2,y2],r[x1,y1]] = ECSTAR; } }

SetNTPPRegion(x1, y1, x2, y2)
{ relations[r[x1,y1],r[x2,y2]] = NTPPISTAR;
  relations[r[x2,y2],r[x1,y1]] = NTPPSTAR; }

```

Figure 3 shows one part-processed image in a sequence captured from the viewpoint of our upper torso humanoid robot *Ludwig*. The objects are each of a distinctive colour and the raw image was captured using a Canon A80 digital camera, then suitably cropped and re-sized, but otherwise unmodified, prior to presentation to the program. The hue bands were adjusted by hand to achieve a clear region segmentation. Figure 4 shows the result of applying the algorithm to the image; note the diagonal symmetry.

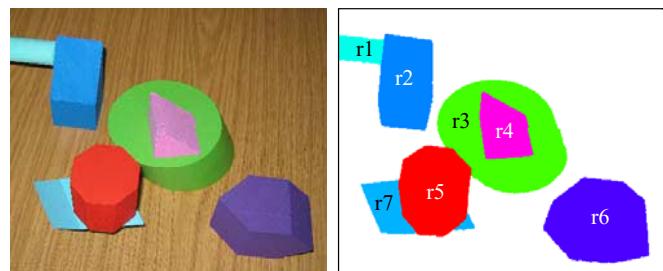


Figure 3: Camera image before and after region extraction

	r1	r2	r3	r4	r5	r6	r7
r1	-	ec*	dc*	dc*	dc*	dc*	dc*
r2	ec*	-	dc*	dc*	dc*	dc*	dc*
r3	dc*	dc*	-	ntppi*	ec*	dc*	dc*
r4	dc*	dc*	ntpp*	-	dc*	dc*	dc*
r5	dc*	dc*	ec*	dc*	-	dc*	ec*
r6	dc*	dc*	dc*	dc*	dc*	-	dc*
r7	dc*	dc*	dc*	dc*	ec*	dc*	-

Figure 4: The resulting detector relations matrix

6 RECOVERING OCCLUSION RELATIONS

We use the *Event Calculus* [14,15], with its primitive ontology of actions (and events), fluents, and time points, for representing and reasoning about occlusion events over time. For the purposes of this paper we will restrict our discussion to the *Happens*(α, t) and the *HoldsAt*(β, t) predicates. Visual occlusion events are modelled as time indexed *fluents* arising from the result of actions, e.g. moving and changing the viewpoint. Thus, for example, where from viewpoint v , body A partially hides body B at time t : this becomes: *HoldsAt*($PH(A, B, v), t$).

As a simple example, consider the sequence where two bodies A and B pass through the *LOS-14* transition path: $C(A, B) \rightarrow JC(A, B) \rightarrow PH(A, B) \rightarrow JF(A, B) \rightarrow F(A, B)$ respectively from times $t0$ to $t4$ – see Figure 1. In the interest of clarity only, we will assume a set of totally ordered time points: $t0$ through to $t4$. We also will assume that throughout this period the bodies are visible. Initially, we infer $C(A, B)$ at $t0$ (from dc*) and $F(A, B)$ (from ntpp*) at $t4$, while all the intermediate states between the time points $t1$ to $t3$ currently remain indistinguishable. This is resolved as follows.

From *Happens*(dc*(image($A, v0$), image($B, v0$), $t0$)) we can abductively infer *HoldsAt*($C(A, B, v0), t0$), and from the change of the detector at $t1$ (with only one direct CND transition) that *HoldsAt*($JC(A, B, v1), t1$) occurred. Given *Happens*(ntpp*(image($A, v4$), image($B, v4$), $t4$)) we know *HoldsAt*($F(A, B, v4), t4$). From the CND only *EH*(A, B) and *JF*(A, B) neighbour *F*(A, B). However, given that during period from $t0$ to $t4$, the detection of both A and B took place, then *EH*(A, B) at $t3$ is ruled out – leaving *HoldsAt*($JF(A, B, v3), t3$). And given it is not possible to pass from *JF* directly to *JC* without passing through *PH* (from the CND), then another event is interpolated, namely: *HoldsAt*($PH(A, B, v2), t2$) thus completing the sequence and satisfying the constraints imposed by the CND.

As for the remaining paths, clearly, parity of reasoning accounts for the path: $C(A, B) \rightarrow JC(A, B) \rightarrow PH(A, B) \rightarrow JF(A, B) \rightarrow Fi(A, B)$. By registering when a body is or is not visible, other sequences can be similarly determined: for example, $PH \rightarrow EH \rightarrow F$ (and its inverse sequence). It turns out that within this dynamic setting, 10 of the 14 relations (interpreted as occlusion events) can now be accounted for, leaving each of the pairs: JH and H ; and JHi and Hi as necessarily indeterminate. Hence, despite the paucity of the visual detectors as to occlusion events, and with minimal modelling assumptions, it becomes possible to infer the occlusion of bodies in three-dimensional space. A similar approach appears in [16] where *Aspect Graphs* replace the role of CNDs.

7 CONCLUSIONS

This paper demonstrates how, within an abductive framework, a rich set of occlusion spatial relations can be reliably inferred using a sparse and simple set of region-relation detectors. The decision to use simple detectors is justified on several grounds. Firstly, the difficult task of providing a clear interpretation for the detection of fine-tuned spatial relationships is now transferred to the inference mechanism. Constraints within the domain theory and associated CND are used in the recovery and justification of the interpolated relation made. Secondly, the sparse set of detectors is robust in use and hence less likely to return hypotheses that will later need to be revised. Thirdly, that the theoretically motivated minimal 3-adjacency neighbourhood property of pixels in segmented regions (and assumed by our algorithm) has additional practical consequence of providing a despeckle and general clean-up operation on raw images. This not only reduces the need to label more potential

regions, it also reduces the need to explain away data not directly supported by the intended model.

We have concentrated on region-contact spatial relations. However, other spatial logics can be similarly grounded in image data to capture more detail in the model. For example, variants on Allen's Interval logic [1] can be used to encode regions being to the left/right or above/below each other - see [10] for an example of this. While *LOS-14* assumes a domain of discrete convex bodies, the segmentation algorithm used does not capture this geometric property. However, by implementing this geometrical property, 7 of the 14 *LOS-14* relations can be directly detected. In the case where convexity condition no longer applies, *ROC-20* can be used to replace *LOS-14* and similarly mapped to the set of detectors – see [10] for the mapping between the relation sets: *ROC-20*, *LOS-14* and *RCC-8*.

ACKNOWLEDGEMENTS

This work was supported by EPRSC under grant: EP/C530683/1(P) “Abductive Robot Perception: Modelling Granularity and Attention in Euclidean Representational Space”, which is gratefully acknowledged. We also wish to thank Murray Shanahan, and to the anonymous referees for detailed comments on an earlier draft of this paper.

REFERENCES

- [1] Allen, J. F. (1983) Maintaining Knowledge about Temporal Intervals, Communications of the ACM, 26(11), pp. 832-843
- [2] Bennett, B. (1994) Spatial Reasoning with Propositional Logic, KR-94, pp. 51-62
- [3] Cohn, A. G. (1997) Qualitative Spatial Representation and Reasoning Techniques, KI-97, Springer LNAI 1303, pp. 1-30
- [4] Freksa, C. (1992) Conceptual neighborhood and its role in temporal and spatial reasoning, in: Singh, M., Trave-Massuyes, L. (eds.) Decision Support Systems and Qualitative Reasoning, North-Holland, Amsterdam, pp. 181-187
- [5] Galton, A. P. (1994) Lines of Sight, in AISB Workshop on Spatial and Spatio-Temporal Reasoning
- [6] Galton, A. P. (1999) The Mereotopology of Discrete Space, in Proc. COSIT 1999, pp. 251-266
- [7] Klette, R. and Rosenfeld, A. (2004) Digital Geometry: Geometric Methods for Digital Picture Analysis, Morgan Kaufmann, Elsevier Inc., San Francisco.
- [8] Kohler, C. (2002), The Occlusion Calculus, in Proc. Workshop on Cognitive Vision, Zurich, Switzerland
- [9] Randell, D. A., Cui, Z. and Cohn A. G. (1992) A Spatial Logic Based on Regions and Connection, KR-92, pp. 165-176
- [10] Randell, D. A., Witkowski, M. and Shanahan, M. (2001) From Images to Bodies: Modelling Spatial Occlusion and Motion Parallax, IJCAI-01, pp. 57-63
- [11] Randell, D. A. and Witkowski, M. (2002) Building Large Composition Tables via Axiomatic Theories, KR-02, pp. 26-36
- [12] Randell, D. A. and Witkowski, M. (2004) Tracking Regions Using Conceptual Neighbourhoods, ECAI-2004 Workshop on Spatial and Temporal Reasoning, pp. 63-71
- [13] Roy, A. J. and Stell, J. G. (2002) A Qualitative Account of Discrete Space, GIScience 2002, pp. 276-290
- [14] Shanahan, M. P. (1997), Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia, MIT Press
- [15] Shanahan, M. P. (1999) The Event Calculus Explained, in Artificial Intelligence Today, Springer LNAI 1600, pp. 409-430
- [16] Shanahan, M. P. and Randell, D. A. (2004) A Logic-Based Formulation of Active Visual Perception, KR-04, pp. 64-72

Abductive Logic Programming in the Clinical Management of HIV/AIDS

Oliver Ray¹ and Athos Antoniades¹ and Antonis Kakas¹ and Ioannis Demetriadis²

Abstract. This paper presents a new Abductive Logic Programming (ALP) approach for assisting clinicians in the selection of anti-retroviral drugs for patients infected with Human Immunodeficiency Virus (HIV). The approach is comparable to laboratory genotypic resistance testing in that it aims to determine which viral mutations a patient is carrying and predict which drugs they are most likely resistant to. But, instead of genetically analysing samples of the virus taken from patients – which is not always practicable – our approach infers likely mutations using the patient’s full clinical history and a model of drug resistance maintained by a leading HIV research agency. Unlike previous applications of abduction, our approach does not attempt to find the “best” explanations, as we can never be absolutely sure which mutations a patient is carrying. Rather, the intrinsic uncertainty of this domain means that multiple alternative explanations are inevitable and we must seek ways to extract useful information from them. The computational and pragmatic issues raised by this approach have led us to develop a new ALP methodology for handling numerous explanations and for drawing predictions with associated levels of confidence. We present our *in-Silico Sequencing System* (iS3) for reasoning about HIV drug resistance as a concrete example of this approach.

1 Introduction

The World Health Organisation estimates 40 million people worldwide are infected with Human Immunodeficiency Virus (HIV). Since there is no cure for this disease, medical intervention aims to postpone the life-threatening collapse of immune function known as Acquired Immune Deficiency Syndrome (AIDS) [15]. Although potent combinations of drugs can slow its progression, the treatment of HIV is complicated by the propensity of the virus to accrue mutations that confer resistance to known medications. Thus clinicians are often faced with the task of designing *salvage therapies* for patients whose first- or second-line treatments are failing.

To improve the likelihood of finding effective salvage therapies medical guidelines advocate the use of laboratory *resistance tests* to help identify which viral mutations a patient is carrying and predict which drugs they are most likely resistant to [8]. But such tests require expert interpretation as they cannot reliably detect *minority* strains of the virus in a patient’s bloodstream or *archived* strains residing in less accessible tissues which may conceal drug resistant mutations. Thus clinicians must carefully analyse a patient’s medical history in order to infer the presence of other likely mutations not revealed by resistance testing.

This paper presents an experimental logic-based approach that can assist in the interpretation of resistance tests by also reasoning with clinical data summarising the success or failure of previous treatments. In particular, we describe a system called *in-Silico Sequencing System* (iS3) that uses Abductive Logic Programming (ALP) [12] to compute sets of mutations that explain a patient’s treatment history. A unique feature of this system is that it can make recommendations even in the absence of resistance tests – which may not be an option for many patients due to prohibitive costs, restricted access or technical limitations. In brief, iS3 uses the same rules as state-of-the-art HIV resistance test interpretation algorithms, but, instead of using them *forwards* to predict likely drug resistances implied by observed mutations, it uses them *backwards* to explain the observed drug response of a patient in terms of likely mutations.

A notable feature of this application domain is that even when resistance tests are available we can never be absolutely sure which mutations a patient is carrying, and so multiple explanations are unavoidable. Thus, in contrast to existing work on abduction, which is aimed at finding the so-called “best” explanations, we must adopt a more pragmatic approach that accepts the plurality of solutions and seeks to extract useful information from them. To this end we propose a three phase method comprising: (i) the development of a heuristic ALP model based on experimentally determined domain rules and prior knowledge, (ii) the introduction of general purpose and domain specific preference criteria to enable the extraction of ranked abductive hypotheses from the multitude of explanations, and (iii) the use of test data to validate the model and to calibrate the rankings by associating a level of confidence with each hypothesis.

The computational challenge of generating and handling large numbers of abductive explanations is considerable as thousands of explanations are typically generated for a single patient. In fact, such computational considerations required us to develop a new abductive engine, described in [16], simply to return these explanations in a feasible time. But, the sheer number of explanations is still too great to allow the efficient extraction of useful information. This motivates the introduction of minimality criteria to reduce the number of redundant explanations considered. Thus, we also describe an efficient method of computing minimal abductive explanations by explaining the observed data incrementally according to the temporal sequence by which it is generated. To demonstrate the utility of our approach, this paper introduces the iS3 drug resistance system as a concrete instance of our general ALP methodology.

The paper is structured as follows. Section 2 introduces the background material for HIV/AIDS and ALP. Sections 3 and 4 describe the ALP model and iS3 approach. Section 5 evaluates the approach using the clinical histories and genotypic data from 10 HIV-infected patients. We conclude with a discussion of related and future work.

¹ University of Cyprus, Cyprus, email: {oliver,athos,antonis}@cs.ucy.ac.cy
² Gregoriou AIDS Clinic, Cyprus, email: idemetriades@mphs.moh.gov.cy

2 Background

2.1 HIV resistance testing

Since its detection in the 1980's, HIV has become a global epidemic. Although several anti-HIV drugs have been developed, they are not very effective in isolation and so infected patients are usually given combination therapies of 3 or 4 drugs to disrupt different stages in the viral life-cycle [15]. Unfortunately, treatment is complicated by the tendency of HIV to develop mutations that confer resistance to these medications [8]. In essence, mutations are copying errors arising in the viral genome that result in amino acid substitutions in the protein sequences coded by those genes. Such changes can reduce the efficacy of certain drugs which target those proteins. For this reason, *resistance tests* are used to help identify which mutations a patient is carrying and predict which drugs they are most likely resistant to. Such tests are of two types [8]: *phenotypic assays* measure the ability of laboratory-cultured clinical isolates to thrive under various concentrations of a particular drug; while *genotypic assays* employ genetic probing or sequencing technologies to identify mutations in the genomes of viruses from infected patients and then apply statistical or rule-based methods to predict levels of drug resistance (based on data from large clinical association studies). However, resistance tests suffer from three limitations [8]. First, they are expensive (250-750\$) and require equipment to which many HIV-infected individuals do not have access. Second, they are ineffective for patients with low blood viral loads ($\leq 500\text{-}900$ copies/ml). Third, they are insensitive to minority strains (comprising $\leq 10\text{-}20\%$ of the viral population) that could lead to the failure of a salvage treatment. Consequently, when they are available, resistance tests require expert interpretation that also takes into account prior clinical history, and, when they are not available, it would be highly desirable to have a means of inferring likely mutations from clinical history alone. To this end, we propose the use of ALP, which is described below.

2.2 ALP Knowledge Representation

Abduction is a form of inference that reasons from effects to causes. Given a goal or observations G and a (logical) theory T , abduction returns explanations Δ that logically entail the goal with respect to the theory. Each explanation is a set of ground atoms whose predicates are specified as *abducible*. Intuitively, the abducibles A are predicates whose extents are incompletely specified by the theory, but which may be subject to further integrity constraints IC .

For computational purposes we use the framework of Abductive Logic Programming (ALP) [12, 6]. Formally, an ALP theory is a triple (T, A, IC) where T is a logic program (the theory), A is a set of predicates (the abducibles) and IC is a set of logical formulae (the integrity constraints). A goal G is a set of literals and an *explanation* of G with respect to (T, A, IC) is a set Δ of ground atoms with predicates in A such that: $T \cup \Delta \models_{LP} G$ and $T \cup \Delta \models_{LP} IC$, where \models_{LP} is some standard entailment relation of logic programming (in this paper we assume the *stable model* semantics).

Each abductive explanation Δ is a hypothesis that together with the model described in T explains how an experimental observable G could hold. The role of the integrity constraints IC , which are modularly stated in the theory, is to augment the basic model in T with any partial information on the abducible predicates or to impose additional validity requirements on the hypotheses Δ .

3 Model of HIV Drug Resistance

Our drug resistance model is automatically extracted from genotypic interpretation rules maintained by the French national AIDS research agency (ANRS) [3]. These rules come from expert analysis of large clinical trials investigating which mutations are associated with resistance to which drugs. We now briefly explain how these associations are represented in our model using the drug *zidovudine* as an example. The ANRS database contains four rules for this drug:

- Z_1 : Mutation Q151M.
- Z_2 : Insertion at codon 69.
- Z_3 : Mutation T215Y/F.
- Z_4 : At least 3 mutations in: M41L, D67N, K70R, L210W, T215A/C/D/E/G/H/I/L/N/S/V, K219Q/E.

Z_1 states that mutation Q151M causes *zidovudine* resistance. The notation Q151M refers to a substitution at codon 151 in the virus where the (wild type) amino acid glutamine, Q, is replaced by (the mutant) methionine, M. In our model, we use atoms of the form *resistant*(P, T, D) to denote that patient P is resistant to drug D at time T , and atoms of the form *mutation*(P, T, M) to denote that patient P is carrying mutation M at time T . Thus we represent Z_1 above by the clause T_1 below. For brevity, we omit the wild types in our formalism – as these are implied by the reference strain – and represent Q151M as 151M. Z_2 states the *insertion* of an amino acid at codon 69 is another indicator of resistance. This is modelled in clause T_2 using "1" to denote an insertion. Z_3 states that either 215Y or 215F will cause resistance. These *alternate* mutations are represented by the term "215YF" in T_3 . Z_4 states that resistance will result if the respective mutations are present at any three of the six codons listed. This is encoded in clause T_4 using the predicate *present*/4 defined in clauses T_5 , T_6 and T_7 to denote the presence in patient P at time T of at least N mutations from the given list. For convenience, the clauses below are all written in standard Prolog notation where variables begin with uppercase letters, the outer universal quantifiers are omitted and commas implicitly denote conjunctions.

- T_1 : *resistant*($P, T, zidovudine$) \leftarrow *mutation*($P, T, "151M"$).
- T_2 : *resistant*($P, T, zidovudine$) \leftarrow *mutation*($P, T, "69i"$).
- T_3 : *resistant*($P, T, zidovudine$) \leftarrow *mutation*($P, T, "215YF"$).
- T_4 : *resistant*($P, T, zidovudine$) \leftarrow *present*($P, T, 3, ["41L", "67N", "70R", "210W", "215ACDEGHILNSV", "219QE"]$).
- T_5 : *present*($P, T, N, [M|Ms]$) \leftarrow *present*(P, T, N, Ms).
- T_6 : *present*($P, T, N, [M|Ms]$) \leftarrow *mutation*(P, T, M), K is $N-1$, *present*(P, T, K, Ms).
- T_7 : *present*($P, T, 0, Ms$).

The ANRS database also contains more complex rules, built from conjunctions of simpler expressions. An example is the rule V_1 for *indinavir*, which is represented in our model by the clause T_8 below.

- V_1 : L90M and at least 2 mutations in: K20M/R, L24I, V32I, M36I, I54V/L/M/T, A71V/T, G73S/A, V77I.
- T_8 : *resistant*($P, T, indinavir$) \leftarrow *mutation*($P, T, "90M"$), *present*($P, T, 2, ["20MR", "24I", "32I", "36I", "54VLMT", "71VT", "73SA", "77I"]$).

For convenience, iS3 automatically downloads the ANRS rules from the Stanford HIV Drug Resistance Database (HIVDB) [19], where they are stored in a more convenient XML format called the Algorithm Specification Interface (ASI) [2]. The XML is parsed by a *document object model* compiler and processed by a *definite clause grammar* in order to extract a logical theory. In this way, we obtain 64 rules for 16 drugs. Currently, we only process rules with in the database which have the highest confidence rating, as these represent the surest indicators of resistance.

Where possible, alternate mutations like 54VLMT are modelled by single terms as this reduces the number of essentially equivalent explanations that need be computed. But, to ensure completeness, it is necessary "break up" alternate mutations that partially overlap with other mutations for the same codon. For example, there are three mutations in the database for codon 54, namely, 54VLMT, 54V and 54AMV, which must be decomposed into the non-overlapping fragments 54V, 54LT, 54M and 54A. For efficiency, alternate mutations are only broken down into the largest possible fragments.

In order to facilitate the computation of minimal abductive explanations some additional equivalence-preserving transformations are applied to the extracted rules such as combining the three rules $T_1 \wedge T_2 \wedge T_3$ into one single rule `resistant(P, T, zidovudine) ← present(P, T, 1, ["151M", "69i", "215YF"])`. As described in the next section, this rewriting is useful because the handling of the predicate `present/4` can be optimised in such a way that reduces the number of non-minimal explanations returned by the abductive proof procedure.

The post-processed theory can be used in two different ways. It can either be used *deductively* to perform standard genotypic test interpretation by asserting known mutations like `mutation(p056, 2, "215YF")` and deriving implied resistances like `resistant(p056, 2, zidovudine)`. More importantly, it can be used *abductively*, for what we call *in-silico* sequencing, by explaining observed resistances in terms of possible mutations. For example, declaring `mutation/3` as the only abducible predicate, there are 23 possible explanations for zidovudine resistance.

This model can now be used to explain the observed ineffectiveness (resp. effectiveness) of a prescribed *set* of drugs by hypothesising the presence (resp. absence) of mutations that imply resistance (resp. non-resistance) to at least one drug in that set. For example, there are 73 explanations for the ineffectiveness of the combination therapy consisting of the three drugs *zidovudine*, *lamivudine* and *indinavir*. Conversely, the effectiveness of a therapy can be explained by the absence of any mutations that would imply resistance to those drugs.

To capture the dynamics of drug resistance, we need an integrity constraint to enforce a key biological assumption in this work: namely, the persistence of mutations. This assumption is motivated by research showing that drug resistant strains of HIV can persist at undetectable levels for long periods (close to the life expectancy of a patient) only to re-emerge when more favourable drugs are prescribed. This constraint, formalised in I_1 below, states that if a patient P carries a mutation M at time T_1 , then he or she will carry that mutation at all later times T_2 .

```
 $I_1: \text{false} \leftarrow \text{mutation}(P, T_1, M), T_2 > T_1,$ 
 $\quad \quad \quad \neg \text{mutation}(P, T_2, M).$ 
```

In summary, our ALP model consists of the rules extracted from the ANRS database, the integrity constraint above and the abducible predicate `mutation/3`. In the next section, we explain how this model is used in iS3 to reason with real patient clinical data.

4 Analysis of Clinical Data

Given a patient's clinical history in the form of facts stating which combinations of drugs were effective or ineffective at given times, iS3 deploys an ALP procedure [16] to infer sets of mutations that explain these observations according to the model in the previous section. The objective is to obtain a set of hypotheses representing different ways of explaining the data which can then be analysed in order derive useful predictions about drug resistance.

As illustrated above, ineffective treatments suggest that certain mutations may have occurred, while effective ones suggest that certain others did not – possibly leading to a violation of integrity that results in the rejection of incorrect explanations. But, for a typical patient – by which we mean one taking 3 different drugs at each of 10 different time points – the computation produces more solutions (many thousands) than can be conveniently analysed.

For this reason, it is necessary to use additional preference criteria to reduce the search space. Assuming that patients adhere to their medications and are regularly monitored to ensure early detection of treatment failure, it is plausible [8] that mutations accrue relatively slowly. We can model this through the notion of *minimality* [12] – which means we only accept explanations from which no smaller explanations can be obtained by removing atoms.

To enforce the minimality of explanations, we optimised the ALP procedure's handling of `present/4` so that, for example, given the subgoal `present(P, T, 1, ["151M", "69i", "215YF"])` it does not consider the first two redundant choices if it has previously established `mutation(P, T, "215YF")`. We also explicitly remove any non-minimal explanations from the solution set obtained after processing each time point. In this way, the number of explanations for the typical patient is cut down to a few hundred.

From this set of explanations explaining a patient's clinical history by the mutations they may or may not be carrying, our task is to analyse these explanations in order to predict for which drugs the patient may be resistant. As a first step, we propose (i) to rank each mutation in the list according to the number and size of explanations it appears in and (ii) to rank each drug in our model according to the number of explanations that imply its resistance.

For task (i), we compute a score for each mutation such that each explanation in which mutation appears contributes a fixed value that decreases with the size of the explanation. (We view this score as simple scalar quantity whose only purpose is to allow the ranking of hypotheses.) The clinical history of a typical patient results in a ranking that contains 30 mutations (out of a possible 60 for which we have rules – or rather 100 if compound mutations are broken up) with scores ranging from 0 to 500, where higher scores indicate a greater likelihood the mutation is present. For example, the top ten ranked mutations for patient p056 are as follows: 215YF (270), 69i (270), 151M (270), 82S (256), 84AV (256), 82T (256), 82AF (256), 46IL (256), 90M (192), 82M (179), etc.

For task (ii), we consider each drug in turn and then count the number of explanations which imply resistance to that drug by essentially using the model in a deductive fashion to perform standard genotypic interpretation on each hypothesised set of mutations. Once again the resulting scores typically range from 0 to 500, with higher scores denoting a greater likelihood of resistance to that drug. For example, the top ten ranked drugs for patient p056 are as follows: efavirenz (252), stavudine (240), zidovudine (230), nevirapine (224), indinavir (81), lamivudine (58), emtricitabine (58), saquinavir (55), ritonavir (53), nelfinavir (49), etc.

5 Calibration and Evaluation

So far we have described a system that, given a patient's clinical history, produces a ranked list of the mutations the patient might be harbouring and the drugs to which they may be resistant. Now we would like to establish some measure of confidence in those lists and identify clinically relevant cut-offs. To do this, we apply the system to data from 10 HIV-infected patients who, over a period of two years, underwent regular medical examinations, routine blood tests, and were genotyped once.³ Some statistics for these patients are shown in the table below. On average, there are six time points for each patient (abstracted in simple way from his treatment history), of which roughly 2 are treatment failures. For reference, the numbers of minimal explanations and ranked mutations are also shown.

patient ID	time points	treatmt. failures	minimal explns.	ranked mutations
p027	9	3	1283	29
p032	5	2	51	32
p056	7	3	333	40
p085	6	3	138	26
p089	9	3	1628	26
p101	7	2	191	32
p113	6	2	1938	38
p114	5	1	73	28
p166	4	1	73	28
p184	6	2	72	27

The only way of assessing whether the mutations predicted by the system are biologically correct is to compare the predicted mutations with the result of the genotype. But this is problematic given that our system is an attempt to overcome the fact genotyping cannot reveal all of the mutations carried by a patient. Thus, while we accept that any mutations detected by the genotype are carried by the patient, we expect that our system will return mutations that are not detected by the genotype but are carried by the patient nonetheless.

Figure 1 (a) plots the percentage of mutations detected by the genotype (on the y-axis) that are contained in the corresponding percentage of top-ranked mutations (on the x-axis). This graph shows that the observed mutations are distributed evenly through the ranking.⁴ In order to obtain a more useful estimate of the system's performance, it is necessary to compensate for selectiveness of the genotype – which is due to the fact that the drugs taken by the patient at the time of genotype reduce non-resistant strains to undetectable levels, while providing the conditions for resistant strains to thrive.

We therefore expect that mutations which are resistant to these drugs will be detected by the genotype with more certainty than those which are not. To account for this, we use our resistance rules to find those mutations that confer no resistance to any of the drugs being taken at the time of the genotype, and we place these mutations further down the ranking while preserving their relative order.

Figure 1 (b) shows the corresponding plot for the adjusted ranking. We see that observed mutations are now ranked much higher, with the majority contained in the top two thirds. While there is insufficient data to draw any significant conclusion⁵, this increases our confidence the system is working correctly, and it suggests a useful

clinical cut-off in the sense that the top 2/3 of the ranked mutations appear to contain most of the useful information.

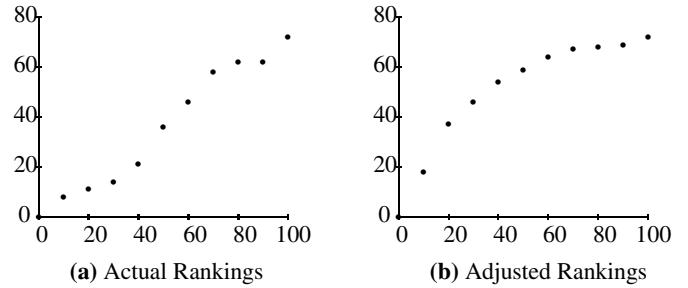


Figure 1. % of genotyped mutations (y) vs. % of top ranked mutations (x)

We would now like to establish whether this 2/3 cut-off also applies to the drug resistances predicted by the system. This is arguably more important since it represents the actual mode of operation for which the system is intended. To do this using the data available to us, we adopted a "leave-one-out" strategy, whereby the system was run on the clinical history of each patient up to *but not including* the last time point with a known (effective or ineffective) outcome at which a new set of drugs was prescribed. We compared the observed effectiveness of the drugs at the next time point with the predicted effectiveness returned by the system.

The results are shown in the table below. For each patient, the 16 drugs are each denoted by one of the symbols '.', '+' or '*' which are distributed in order of decreasing predicted resistance from left to right. A plus denotes a drug contained in the treatment at next time point. Because the prescribed treatments were all found to be effective at the following time point, we would like for the system to give these drugs a low resistance. As shown in the table, they were mostly predicted medium or low resistance, indicating a better than average performance. A star denotes a drug for which resistance can be inferred using the rules of the system and the mutations in the genotype. We would like the system to rank these highly. As shown below, this is generally the case, once again suggesting a better than average performance. All other drugs are denoted by a dot.

Interestingly, half of the patients were prescribed one drug with high predicted resistance. But, in all cases, the other drugs were ranked much lower – which is consistent with the overall success of the treatment. The system performed very well for p166, where resistance to the two top-ranked drugs was independently supported by the genotypic evidence, and two of the drugs found to be effective at the next time point received a low or medium prediction.

Patient ID	Predicted Drug Resistance		
	high	medium	low
p027++..+..
p032	.+.**	+.*..+..
p056	+...+*	+....+
p085	+....	.+....	..+...+
p089	+....	.+....+..
p101	.+...	...++.	+.....
p113+	..++..
p114++..+.*
p166	*+..+	...++.	..+....
p184	..+..	.+....

+ drugs used in effective therapy at next time point

* drugs inferred to be ineffective from genotype data

³ Unfortunately, clinical data from 10 additional patients was considered too unreliable by an HIV specialist to include in this study.

⁴ The graph only rises to 80% as some genotyped mutations are not ranked. These could be given a default score of 0, but this serves no real purpose.

⁵ The genotype only returns results for 33 of 97 mutations in our rules, so that the presence or absence of most ranked mutations is unknown.

6 Discussion and Related Work

Previous work on HIV resistance is aimed at predicting drug resistance from the results of genotypic or phenotypic tests performed on clinical isolates from infected patients [1]. Our work does just the opposite, by explaining previously observed resistance in terms of underlying mutations and using these explanations to predict possible drug resistance. Our model of HIV drug resistance is derived from the ANRS genotypic interpretation algorithm [3]. Other rule-based interpretation algorithms could have been used instead, such as those compared in [23] or those available from the Stanford HIVDB web site [19], but ANRS was selected because it was more amenable to the extraction of logical rules.

Previous applications of abduction, such as natural language understanding [9], planning [20, 11], perception [21] and diagnosis [4, 17, 7], take the approach of inference to the "best explanation" [10] by using domain specific constraints to identify one or more optimal hypotheses. The (often implicit) assumption in these applications is that we have sufficient information to discriminate between alternative explanations. Some applications such as [13] and [5] even combine abduction and constraint solving in order to solve the implicit optimisation problems underlying these domains. When this assumption does not hold it is suggested that we carry out "crucial experiments" [18] to obtain further information that would help home-in on the optimal hypotheses.

Our work offers an alternative approach to real-life applications in which the underlying model does not (and cannot) provide enough information to reliably discriminate between competing hypotheses. In these cases, we suggest the emphasis should be placed on extracting useful information from the multitude of explanations and helping the user to make informed decisions. This view is particularly relevant in applications of abduction to scientific theory formation, such as [24, 14, 22], where the task is to improve a necessarily incomplete model of the problem domain.

7 Conclusions and Future Work

This paper introduced a novel AI approach for abductively inferring mutations carried by HIV-infected patients and predicting likely drug resistance from clinical data. The proposed approach is a concrete instance of a new ALP methodology for addressing the issue of multiple explanations through a process of ranking abductive hypotheses extracted from the multitude of explanations and empirically calibrating the system to obtain a measure of confidence in its results. Our in-silico sequencing system, iS3, is a practical implementation with the potential to assist clinicians in the selection of anti-retroviral drugs – especially in cases where laboratory resistance tests are not available. The system automatically keeps itself updated with the latest drug resistance data provided by a leading HIV research body.

We are currently preparing the system for a trial deployment in an HIV/AIDS clinic where we hope to assess the clinical utility of the system and carry out further tests on another cohort of HIV carriers from a different geographical areas. We are also investigating possible improvements to the system by (i) extending the domain model with additional information about common *pathways* by which some viral mutations are known to occur, by (ii) incorporating information about *antagonistic mutations* that partially reverse the resistance effects of some other mutations, and by (iii) exploiting statistical information associated with the drug resistance rules used in the model. We are also keen to examine how our general ALP approach for handling multiple explanations can be applied to other applications.

ACKNOWLEDGEMENTS

We are grateful to Leondios Kostrikis, who helped with the interpretation of the genotypic tests, and to Bradely Betts for his assistance with the Stanford Algorithm Specification Interface. We also thank the anonymous referees for their helpful suggestions.

REFERENCES

- [1] N. Beerenwinkel et al., 'Computational methods for the design of effective therapies against drug resistant HIV strains', *Bioinformatics*, **21**(21), 43943–3950, (2005).
- [2] B.J. Betts and R.W. Shafer, 'Algorithm Specification Interface for Human Immunodeficiency Virus Type 1 Genotypic Interpretation', *Journal of clinical microbiology*, **41**(6), 2792–2794, (2003).
- [3] F. Brun-Vezinet, B. Masquelier et al. Agence Nationale de Recherche contre le Sida - AC11 resistance group - genotypic drug resistance interpretation rules (v13), 2005. At <http://pugliese.club.fr/index.htm>.
- [4] L. Console et al., 'Using compiled knowledge to guide and focus abductive diagnosis.', *IEEE Transactions on Knowledge and Data Engineering*, **8**(5), 690–706, (1996).
- [5] M. Denecker et al., 'Scheduling by "abductive execution" of a classical logic specification', in *Proceedings of the ERCIM/Compulog workshop on Constraints*, pp. 1–5, (1997).
- [6] M. Denecker and A.C. Kakas, *Abduction in Logic Programming*, In Computational Logic: From Logic Programming into the Future, Springer-Verlag, 2001.
- [7] J. Gartner et al., 'Psychiatric diagnosis from the viewpoint of computational logic', in *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pp. 1362–1376. Springer-Verlag, (2000).
- [8] M.S. Hirsch et al., 'Antiretroviral Drug Resistance Testing in Adults Infected with Human Immunodeficiency Virus Type 1: 2003 Recommendations of an International AIDS Society-USA Panel', *Clinical Infectious Diseases*, **37**, 113–141, (2003).
- [9] J.R. Hobbs et al., 'Interpretation as abduction', *Artificial Intelligence*, **63**(1-2), 69–142, (1993).
- [10] J. R. Josephson and S. G. Josephson, *Abductive Inference: Computation, Philosophy, Technology*, Cambridge University Press, 1994.
- [11] A.C. Kakas et al., 'ACLP: Abductive Constraint Logic Programming', *Journal of Logic Programming*, **44**(1-3), 129–177, (2000).
- [12] A.C. Kakas, R.A. Kowalski, and F. Toni, 'The role of abduction in logic programming', in *Handbook of Logic in AI and Logic Programming*, volume 5, 235–324, O.U.P., (1998).
- [13] A.C. Kakas and A. Michael, 'An abductive-based scheduler for air-crew assignment.', *Applied Artificial Intelligence*, **15**(3), 333–360, (2001).
- [14] R.D. King et al., 'Functional genomic hypothesis generation and experimentation by a robot scientist', *Nature*, **427**, 247–252, (2004).
- [15] Panel on Clinical Practices for Treatment of HIV Infection - US Department of Health and Human Services, *Guidelines for the Use of Antiretroviral Agents in HIV-1-Infected Adults and Adolescents*, 2005.
- [16] O. Ray and A.C. Kakas, 'ProLogICA: a practical system for Abductive Logic Programming', in *Proceedings of the 11th International Workshop on Non-monotonic Reasoning*, (to appear 2006).
- [17] A. Sattar and R. Goebel, 'On the efficiency of logic-based diagnosis', in *Proceedings of the 3rd international conference on industrial and engineering applications of AI and expert systems*, pp. 23–31, (1990).
- [18] A. Sattar and R. Goebel, 'Using crucial literals to select better theories', *Computational Intelligence*, **7**(1), 11–22, (1991).
- [19] R.W. Shafer et al. Stanford University HIV Drug Resistance Database, 2005. At <http://hivdb.stanford.edu/index.html>.
- [20] M. Shanahan, 'An abductive event calculus planner', *Logic Programming*, **44**(1-3), 207–239, (2000).
- [21] M. Shanahan, 'Perception as abduction: Turning sensor data into meaningful representation', *Cognitive Science*, (to appear 2006).
- [22] N. Tran, C. Baral, V. Nagaraj, and L. Joshi, 'Knowledge-based framework for hypothesis formation in biochemical networks', *Bioinformatics*, **21**(supplement 2), 213–219, (2005).
- [23] M. Zazzi et al., 'Comparative evaluation of three computerized algorithms for prediction of antiretroviral susceptibility from HIV type 1 genotype', *Antimicrobial Chemotherapy*, **53**(2), 356–360, (2004).
- [24] B. Zupan et al., 'Genepath: a system for automated construction of genetic networks from mutant data', *Bioinformatics*, **19**, 383–389, (2003).

Interleaving belief updating and reasoning in abductive logic programming

Fariba Sadri and Francesca Toni¹

Abstract. Most existing work on knowledge representation and reasoning assumes that the updating of beliefs is performed off-line, and that reasoning from the beliefs is performed either before or after the beliefs are changed. This imposes that, if an update occurs while reasoning is performed, reasoning has to be stopped and re-started anew so that the update is taken into account, with an obvious wastage of reasoning effort. In this paper, we tackle the problem of performing belief updating on-line, while reasoning is taking place by means of an abductive proof procedure.

1 INTRODUCTION

Traditionally, the area of knowledge representation and reasoning has focused on identifying techniques for drawing correct conclusions efficiently from beliefs. Beliefs can be held by intelligent agents, for example, and reasoning can aim at identifying plans for the agents' goals or answering queries on the agents' beliefs. Such beliefs can be subject to updates. The updates can be generated by a number of different means, for example by learning, or, in a multi-agent setting, by observation or by communication amongst agents.

If the updating of beliefs takes place while reasoning is performed, the problem arises as to which parts of the already performed reasoning can be "saved", because it has used beliefs which have not been affected by the update. Conventional knowledge representation and reasoning systems assume that no parts of reasoning can be saved, by restarting the reasoning process anew.

In this paper, we show how belief updating and reasoning can be interleaved, so that reasoning with beliefs that are not affected by the updates can be saved and, if necessary, used after the updating. We assume that reasoning is performed by means of the abductive proof procedure IFF [3] and that beliefs are represented by means of abductive logic programs (ALPs) [6]. ALPs consist of logic programs and integrity constraints. We also assume that belief updating amounts to the addition or deletion of rules in the logic program component of the ALP, or to the addition or deletion of integrity constraints in the corresponding component of the ALP.

In this paper, we do not address how the updates are determined. In particular, we do not focus on "belief revision" (e.g. as understood in [1]), but rather on how updates produced by such revision can be interleaved with the reasoning process.

In order to save any reasoning effort not affected by updates, we adopt a labeling technique that maintains dependencies. We define a new variant of the IFF procedure, called *LIFF*, with labels occurring in derivations. We formalise the soundness of LIFF, wrt the semantics

underlying the IFF procedure. We concentrate on the propositional case only for the definition of LIFF and the results.

Our work is closely related to the work of [4, 5], which share our aims. However, we extend an abductive proof procedure (IFF) for beliefs in the form of ALPs whereas [4] extends conventional SLDNF, for ordinary logic programs. Our knowledge representation and reasoning choice paves the way to the use of our techniques for agent applications, following the abductive logic agent approach of [8, 13, 7]. Moreover, we use a labeling technique rather than the (arguably more space consuming) technique based on full data structures associated to atoms as in [4].

Amongst agent applications for which our approach is particularly useful are those where the agents are situated in dynamic environments and need to adjust quickly to the changes in that environment. For example, a situated pro-active agent can plan for its goals while it interleaves the planning with action execution and observations in the environment; such an agent has, for example been described in [10]. The observations that it makes and records from the environment and the success or failure of the actions that it attempts to execute can lead to the agent revising its beliefs. LIFF allows the agent to keep as much of its partial plan as possible and to replan only those parts that are affected by the changes in its beliefs.

Another agent application that would benefit from our work is information integration via a mediator, for example to realise the semantic web. In this application a mediator agent receives the user query specified in some user-oriented high-level language. It then constructs a query plan translating the user query into a form understandable by the sources of information it has at its disposal. It contacts the sources for the necessary information, integrates their answers, and translates the answer into the language of the user. Such an information integration approach based on abductive logic programming and, in particular the IFF proof procedure, has been described in [19]. Now, while the agent is constructing the query plan for the user query, it may receive information about content or availability changes to the information sources. LIFF would allow the agent to accommodate these changes within its reasoning without having to discard all the work it has done in constructing the query plan.

This paper is a revised and extended version of [12].

2 PRELIMINARIES

Abductive logic programming is a general-purpose knowledge representation and reasoning framework that can be used for a number of applications and tasks [6, 8, 11, 14, 13, 17, 18]. It relies upon a logic-based representation of beliefs, for any given domain of application, via *abductive logic programs* (ALPs), and the execution of logic-based reasoning engines, called *abductive proof procedures*,

¹ Department of Computing, Imperial College London, UK,
 email:{fs, ft}@doc.ic.ac.uk

for reasoning with such representations. In this paper we concentrate on the propositional case.

An ALP consists of

- A **logic program**, P , namely a set of if-rules of the form $\text{Head} \leftarrow \text{Body}$, where Head is an atom and Body is either *true* (in which case we write the if-rule simply as Head) or a conjunction of literals. P is understood as a (possibly incomplete) set of beliefs.
- A set of **abducible atoms**, \mathcal{A} , which are assumed not to occur in the Head of any if-rule in P . Abducible atoms can be used to “augment” the (beliefs held within the) logic program, subject to the satisfaction of the integrity constraints (see below).
- A set of **integrity constraints**, I , namely if-then-rules of the form $\text{Condition} \Rightarrow \text{Conclusion}$, where Condition is either *true* (in which case we write the if-then-rule simply as Conclusion) or a conjunction of literals, and Conclusion is *false* or an atom. The integrity constraints are understood as properties that must be “satisfied” by any “acceptable” extension of the logic program by means of abducible atoms.

In the sequel, $\overline{\mathcal{A}}$ will refer to the complement of the set \mathcal{A} wrt the set of all atoms in the vocabulary of P , i.e. $\overline{\mathcal{A}}$ is the set of non-abducible atoms. Both IFF and LIFF rely upon the **completion** [2] of logic programs. The completion of an atom p defined, within a propositional logic program, by the if-rules $p \leftarrow D_1, \dots, p \leftarrow D_k$ is the **iff-definition** $p \leftrightarrow D_1 \vee \dots \vee D_k$. The completion of an atom p not defined in a given logic program is $p \leftrightarrow \text{false}$. The **selective completion** $\text{comp}_{\overline{\mathcal{A}}}(P)$ of a logic program P wrt a set of atoms S in the vocabulary of P is the union of the completions of all the atoms in S . Both IFF and LIFF use $\text{comp}_{\overline{\mathcal{A}}}(P)$, i.e. the selective completion of P wrt $\overline{\mathcal{A}}$.

Abductive proof procedures aim at computing “explanations” (or “abductive answers”) for “queries”, given an ALP representing knowledge about some underlying domain. A **query** is a (possibly empty) conjunction of literals. Given an ALP $\langle P, \mathcal{A}, I \rangle$, an **explanation** (or **abductive answer**) for a query Q is a (possibly empty) set E of abducible atoms such that $P \cup E \text{ entails } Q$ and $P \cup E \text{ satisfies } I$. Various notions of entailment and satisfaction can be adopted, for example entailment and satisfaction could be entailment and consistency, respectively, in first-order, classical logic. In this paper, we adopt truth wrt the 3-valued completion semantics of [9] as the underlying notion of entailment and satisfaction, inherited from IFF [3].

IFF generates a **derivation** consisting of a **sequence of goals**, starting from an **initial goal**, which is the given query conjoined with the integrity constraints in I . IFF computes explanations (referred to as **extracted answers**) for the given query extracting them from disjuncts in a goal in a derivation from the initial goal, when no further inference rule can be applied to these disjuncts.

Goals in a derivation are obtained by applying **inference rules** (see section 5.1). In the simplest case, these goals are disjunctions of **simple goals**, which are conjunctions of the form

$$A_1 \wedge \dots \wedge A_n \wedge I_1 \wedge \dots \wedge I_m \wedge D_1 \wedge \dots \wedge D_k$$

where $n, m, k \geq 0$, $n + m + k > 0$, the A_i are atoms, the I_i are **implications**, with the same syntax as integrity constraints, and the D_i are disjunctions of conjunctions of literals and implications. Implications are obtained by applying the inference rules of the proof procedure to either integrity constraints in the given $\langle P, \mathcal{A}, I \rangle$ or to the result of rewriting negative literals $\text{not } A$ as $A \Rightarrow \text{false}$.

IFF assumes that the inference rules are applied in such a way that every goal in a derivation is a *disjunction of simple goals*. LIFF will make the same assumption.

We omit here the definition of the inference rules of IFF, as they

can be re-constructed from the inference rules of LIFF, given in section 5, by dropping the labels. In section 4 we illustrate the behaviour of IFF and LIFF with an example. The example illustrates the main difference between IFF and LIFF, namely the fact that the latter associates labels to literals and implications in goals, to be exploited when the ALP is updated.

3 ASSIMILATING UPDATES IN THE ABDUCTIVE LOGIC PROGRAM

As mentioned in section 1, the decision of what to revise and how can be done in a number of different ways. However the belief revision is handled, at the end of the process, if-rules or integrity constraints will need to be added or deleted from the ALP. In this section we define how a given ALP $\langle P, \mathcal{A}, I \rangle$ is revised after (any number of) the following kinds of updates:

- the addition to P of if-rules,
- the deletion from P of if-rules,
- the addition to I of integrity constraints,
- the deletion from I of integrity constraints.

Note that *modification* of if-rules and integrity constraints can be achieved by suitable combinations of deletions and additions.

There are a number of different ways that we can accommodate addition and deletion of if-rules in an ALP. For example, we can decide whether or not an atom that is originally abducible will, in effect, remain abducible after any updates. Similarly we can decide whether or not an atom that is originally non-abducible will always be non-abducible, even if all its definitions are deleted from the logic program. LIFF is independent of these choices, and it can deal uniformly with any combinations or variations of these choices. Below, we arbitrarily make the concrete choice where, after any updates, non-abducible atoms remain non-abducible and abducible atoms remain “abducible”, in the sense that they can always be made to hold by abduction, as we will see later. Below, $\langle P, \mathcal{A}, I \rangle$ denotes the ALP before the update.

3.1 Addition of if-rules

Let the update U be the addition of the if-rule $p \leftarrow B$, with $B \neq \text{false}$,² and with B possibly *true*. We will refer to \mathcal{A}_B as the set of atoms occurring in B but not in the vocabulary of $\langle P, \mathcal{A}, I \rangle$.

We refer to the revision of $\langle P, \mathcal{A}, I \rangle$ by U as $U(\langle P, \mathcal{A}, I \rangle) = \langle P', \mathcal{A}', I \rangle$, obtained as follows:

1. if p is an abducible in \mathcal{A} , then
 - $P' = P \cup \{p \leftarrow B\} \cup \{p \leftarrow p^*\}$,
 - $\mathcal{A}' = ((\mathcal{A} \cup \{p^*\}) - \{p\}) \cup \mathcal{A}_B$,

where p^* is an atom not in the vocabulary of $\langle P, \mathcal{A}, I \rangle$;
2. if p is in the vocabulary of $\langle P, \mathcal{A}, I \rangle$ but is not abducible or p is not in the vocabulary of $\langle P, \mathcal{A}, I \rangle$, then
 - $P' = P \cup \{p \leftarrow B\}$,
 - $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_B$.

Note that, in case 1, the definition of p in P' conforms to our policy that atoms that are abducible in the initial ALP always remain

² The addition of $p \leftarrow \text{false}$ is not allowed directly, but it can be achieved by deleting all if-rules with head p . We assume that p remains in the language of the ALP even after all if-rules for p are deleted from it.

“abducible”, in the sense that they can be assumed to hold by abduction (of atoms p^*). Note also that we assume that any atom in the body of any added rule but not present in the vocabulary of the ALP prior to the update is treated as a new abducible (in \mathcal{A}_B). This choice sees new undefined atoms as “open”. Finally, note that we do not allow the addition of new abducible atoms explicitly, from the outside. However, new abducible atoms are added as a by-product of adding definitions for abducible atoms (case 1 above) or definitions of atoms containing the new abducible atoms in the body, as atoms that did not occur before in the ALP (cases 1-2 above).

3.2 Deletion of if-rules

Let update U be the deletion of $p \leftarrow B \in P$ (B possibly true). We will assume that B is different from a p^* atom, namely an atom introduced in case 1 in section 3.1. Revising $\langle P, \mathcal{A}, I \rangle$ by U gives $U(\langle P, \mathcal{A}, I \rangle) = \langle P', \mathcal{A}, I \rangle$, obtained as follows: $P' = P - \{p \leftarrow B\}$.

3.3 Addition of integrity constraints

Let update U be the addition of $C \Rightarrow D$. Let \mathcal{A}_U be the set of all atoms occurring in C, D but not in the vocabulary of $\langle P, \mathcal{A}, I \rangle$. Revising $\langle P, \mathcal{A}, I \rangle$ by U gives $U(\langle P, \mathcal{A}, I \rangle) = \langle P, \mathcal{A}', I' \rangle$, obtained as follows:

- $I' = I \cup \{C \Rightarrow D\}$;
- $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_U$.

3.4 Deletion of integrity constraints

Let update U be the deletion of $C \Rightarrow D \in I$. Revising $\langle P, \mathcal{A}, I \rangle$ by U gives $U(\langle P, \mathcal{A}, I \rangle) = \langle P, \mathcal{A}, I' \rangle$, obtained as follows: $I' = I - \{C \Rightarrow D\}$.

3.5 Multiple updates

Here we define the effect of multiple updates upon an initially given ALP $\langle P, \mathcal{A}, I \rangle$. Let the sequence of updates be U_1, \dots, U_n , with $n > 1$. Then, U_1, \dots, U_n applied to $\langle P, \mathcal{A}, I \rangle$ gives

$U_n \circ U_{n-1} \circ \dots \circ U_1(\langle P, \mathcal{A}, I \rangle) = U_n(U_{n-1}(\dots(U_1(\langle P, \mathcal{A}, I \rangle))))$. Later, in section 5.2, we will allow empty updates to occur in sequences of updates. An empty update stands for no update at all.

4 AN ILLUSTRATIVE EXAMPLE

In this section we illustrate the application of LIFF and its relationship to IFF via a concrete example. The example also illustrates the potential saving in re-computation, for example, compared to “backtracking” to an appropriate place in the search space.

Given the ALP $\langle P, \mathcal{A}, I \rangle$ with

$$\begin{aligned} P: p &\leftarrow r, \quad q \leftarrow s, \quad r \leftarrow t, \quad m \leftarrow \neg a \\ \mathcal{A}: a, n, s, t \\ I: s \wedge m &\Rightarrow n \end{aligned}$$

and a query $p \wedge q$, IFF may derive the following sequence of goals (where, with an abuse of notation, I will stand for $s \wedge m \Rightarrow n$):

$$\begin{aligned} p \wedge q &\wedge I \\ \text{by unfolding } p \text{ with } \text{Comp}_{\overline{\mathcal{A}}}(P): \\ r \wedge q &\wedge I \\ \text{by unfolding } q \text{ with } \text{Comp}_{\overline{\mathcal{A}}}(P): \\ r \wedge s \wedge I \\ \text{by unfolding } r \text{ with } \text{Comp}_{\overline{\mathcal{A}}}(P): \end{aligned}$$

$$t \wedge s \wedge I$$

by propagation with $s \wedge m \Rightarrow n \in I$:

$$t \wedge s \wedge (m \Rightarrow n) \wedge I$$

by unfolding m with $\text{Comp}_{\overline{\mathcal{A}}}(P)$:

$$t \wedge s \wedge (\neg a \Rightarrow n) \wedge I$$

by negation elimination and splitting:

$$[t \wedge s \wedge a \wedge I] \vee [t \wedge s \wedge n \wedge I]$$

From the final goal in this derivation, IFF would extract the explanations $\{t, s, a\}$ (from the first disjunct) and $\{t, s, n\}$ (from the second disjunct) for the given query.

Given the same $\langle P, \mathcal{A}, I \rangle$ and query, the analogous LIFF derivation would consist of the following sequence of goals with labels. Labels are formally given in definition 1. A label $\{\langle X_1; Y_1 \rangle, \dots, \langle X_n; Y_n \rangle\}$ associated with a literal conjunct or an implication Z in a goal, intuitively indicates that “if X_i is updated, then Z should be replaced by Y_i ”. In the simplest case, each X_i is an atom or an integrity constraint and each Y_i is an atom or an implication or \emptyset , standing for empty. For ease of reading where the label of a conjunct does not change from one goal to the next in the sequence we do not repeat the label.

$$p : \{\} \wedge q : \{\} \wedge I : \{\langle I; \emptyset \rangle\}$$

by unfolding p with $\text{Comp}_{\overline{\mathcal{A}}}(P)$:

$$r : \{\langle p; p \rangle\} \wedge q \wedge I$$

by unfolding q with $\text{Comp}_{\overline{\mathcal{A}}}(P)$:

$$r \wedge s : \{\langle q; q \rangle\} \wedge I$$

by unfolding r with $\text{Comp}_{\overline{\mathcal{A}}}(P)$:

$$t : \{\langle p; p \rangle, \langle r; r : \{\langle p; p \rangle\} \rangle\} \wedge s \wedge I$$

by propagation (with $s \wedge m \Rightarrow n \in I$):

$$t \wedge s \wedge (m \Rightarrow n) : \{\langle q; \emptyset \rangle, \langle I; \emptyset \rangle\} \wedge I$$

by unfolding m with $\text{Comp}_{\overline{\mathcal{A}}}(P)$:

$$t \wedge s \wedge (\neg a \Rightarrow n) : \text{label} \wedge I$$

where label is $\{\langle q; \emptyset \rangle, \langle I; \emptyset \rangle, \langle m; (m \Rightarrow n) : \{\langle q; \emptyset \rangle, \langle I; \emptyset \rangle\} \rangle\}$

by negation elimination and splitting:

$$[t \wedge s \wedge a : \text{label} \wedge I] \vee [t \wedge s \wedge n : \text{label} \wedge I]$$

Suppose at this stage of the derivation there is an update (addition or deletion) to the definition of p . If we were to simply “backtrack” to where the definition of p was used the first time we would go back to the very first goal of the derivation, namely $p \wedge q \wedge I$ and, effectively, start all over again. But using the labels of LIFF we would simply rewrite the last goal of the derivation as $[p \wedge s \wedge a] \vee [p \wedge s \wedge n]$ (for ease of reading we have dropped the labels here) thus in effect, saving the work done in the application of 4 out of the 6 inferences.

5 THE LIFF PROOF PROCEDURE

Definition 1 A label (for an atom or implication) is a set

$$\{\langle X_1; Y_1 \rangle, \dots, \langle X_n; Y_n \rangle\}$$

$n \geq 0$, with each X_i an atom or an integrity constraint and each Y_i either

- empty: \emptyset , or
- an atom (possibly with a label), or
- an implication (possibly with a label), and $X_i \neq X_j$ for all $i \neq j$.

Note that labels can be empty (i.e. if $n = 0$). Each $\langle X_i; Y_i \rangle$ in a label for Z indicates that Z should be replaced by Y_i if any update is made upon X_i . In effect, this label identifies the components that have contributed to the derivation of Z . We impose that all X_i are different so that there is no ambiguity as to what Z has to be replaced with when X_i is updated.

We will refer to labeled atoms/implications simply as atoms/implications. Moreover, the terminology of goals and simple goals will carry through in the presence of labels.

5.1 LIFF inference rules

Given an ALP $\langle P, \mathcal{A}, I \rangle$ and an initial query Q , we will define *LIFF derivations* for Q in terms of sequences of goals, G_1, \dots, G_n , such that $G_1 = Q : \{\} \wedge I^*$, where $(A_1 \wedge \dots \wedge A_k) : 1$ ($k > 1$) stands for $A_1 : 1 \wedge \dots \wedge A_k : 1$ ³, $I^* = \{i : \langle i; \emptyset \rangle | i \in I\}$, and each G_{i+1} is obtained from the previous goal G_i either by enforcing an update (see section 5.2) or by application of one of the inference rules below.

Unfolding an atomic conjunct: given $p \leftrightarrow D_1 \vee \dots \vee D_n$ in $\text{comp}_{\overline{\mathcal{A}}}(P)$ and an atom $p : 1$ which is a conjunct of a simple goal in G_i ,

then G_{i+1} is G_i with $p : 1$ replaced by $(D_1 \vee \dots \vee D_n) : 1'$, where $1' = 1$ if $\langle p; S \rangle \in 1$, for some S , and $1' = 1 \cup \{\langle p; p : 1 \rangle\}$ otherwise.

Unfolding an atom in the conditions of an implication: given $p \leftrightarrow D_1 \vee \dots \vee D_n$ in $\text{comp}_{\overline{\mathcal{A}}}(P)$ and an implication $[L_1 \wedge \dots \wedge L_i \wedge \dots \wedge L_m \Rightarrow A] : 1$ which is a conjunct of a simple goal of G_i with $L_i = p$,
then G_{i+1} is G_i with the implication replaced by the conjunction $[L_1 \wedge \dots \wedge D_1 \wedge \dots \wedge L_m \Rightarrow A] : 1' \wedge \dots \wedge [L_1 \wedge \dots \wedge D_n \wedge \dots \wedge L_m \Rightarrow A] : 1'$, where $1' = 1$ if $\langle p; S \rangle \in 1$, for some S , and $1' = 1 \cup \{\langle p; L_1 \wedge \dots \wedge L_m \Rightarrow A : 1 \rangle\}$ otherwise.

Propagation: given an atom $p : 1$ and an implication $[L_1 \wedge \dots \wedge L_i \wedge \dots \wedge L_m \Rightarrow A] : 1'$ with $L_i = p$, both conjuncts of the same simple goal in G_i ,
then, if the implication $[L_1 \wedge \dots \wedge L_{i-1} \wedge L_{i+1} \wedge \dots \wedge L_m \Rightarrow A] : \{\langle q; \emptyset \rangle | \langle q; r \rangle \in 1 \cup 1'\}$ is not already a conjunct of the simple goal, then G_{i+1} is G_i with the new implication conjoined to the simple goal.

Negation elimination: given an implication $\text{not } A_1 \wedge \dots \wedge \text{not } A_m \Rightarrow A : 1$ which is a conjunct of a simple goal in G_i , then G_{i+1} is G_i with the implication replaced by the disjunction $[A \vee A_1 \vee \dots \vee A_m] : 1$.⁴

Logical simplification replaces:

- $[B : 1 \vee C : 1'] \wedge E : 1''$ by $[B : 1 \wedge E : 1''] \vee [C : 1' \wedge E : 1'']$ (**splitting**)
- $B : 1 \wedge B : 1$ by $B : 1$
- $B : 1 \vee B : 1$ by $B : 1$
- $\text{not } A : 1$, where A is an atom, by $(A \Rightarrow \text{false}) : 1$

Note that we are not including some simplification steps present in IFF, e.g. $A \wedge \text{false} \equiv \text{false}$ and $A \vee \text{false} \equiv A$. Such simplification steps only affect the “efficiency” of IFF, rather than its correctness. We leave these steps out in LIFF because it simplifies the labels while allowing us to incorporate updates and revise the goals when necessary (e.g. revising $A \wedge \text{false}$, if what led to false is updated). For similar reasons we also do not include a simplification step for “merging” identical conjuncts which have different labels.

Note that for goals in LIFF derivations to be guaranteed to be disjunctions of simple goals, it is sufficient that every step of unfolding and negation elimination is followed by a step of splitting. We will assume this is the case in all derivations.

³ In the sequel, $(D_1 \vee \dots \vee D_n) : 1$ will similarly stand for $D_1 : 1 \vee \dots \vee D_n : 1$.

⁴ In [3], negation elimination is defined as follows: $\text{not } A_1 \wedge \text{Rest} \Rightarrow A$ is replaced by $\text{Rest} \Rightarrow A \vee A_1$. Operationally, our definition (ignoring the labels) is equivalent to the one in [3].

5.2 LIFF derivations

In LIFF derivations, the application of inference rules (as given in section 5.1) is interleaved with the assimilation of updates within the ALP (as given in section 3) and the application of these updates to goals, defined as follows:

Definition 2 Given a goal G and an update U , the **updated goal wrt G and U** is a goal G' obtained as follows:

- if U is the addition or deletion of an if-rule $p \leftarrow B$, then G' is G where every conjunct with $\langle p; Q \rangle$ in its label is replaced by Q ;
- if U is the deletion of an integrity constraint X then G' is G where every conjunct with label containing $\langle X; Q \rangle$ is replaced by Q (Q will be \emptyset in all such cases);
- if U is the addition of an integrity constraint i then G' is G with an added conjunct $i : \{\langle i; \emptyset \rangle\}$.

In all cases, any resulting conjunct which is \emptyset is deleted.

Definition 3 Given a query Q , an ALP $\langle P, \mathcal{A}, I \rangle$, and a sequence of updates U_1, \dots, U_n , $n > 0$, a **LIFF derivation** is a sequence

$$(G_1, U_1, R_1, ALP_1), \dots, (G_n, U_n, R_n, ALP_n)$$

where, for each $1 \leq i \leq n$,

- G_i is a goal
- U_i is an update (possibly empty)
- R_i is empty or (an instance of) an inference rule applied to a simple goal in G_i which does not have false as a conjunct
- exactly one of U_i and R_i is non-empty
- ALP_i is an ALP

and $G_1 = Q : \{\} \wedge I^*$ (where I^* is defined in section 5.1), $ALP_1 = \langle P, \mathcal{A}, I \rangle$, and, for each $1 < i \leq n$,

- if R_{i-1} is non-empty then
 - G_i is obtained from G_{i-1} by applying R_{i-1}
 - $ALP_i = ALP_{i-1}$
- if U_{i-1} is non-empty then
 - G_i is the updated goal wrt G_{i-1} and U_{i-1}
 - $ALP_i = U_{i-1}(ALP_{i-1})$.

The **end goal** and **end ALP** in a LIFF derivation

$$(G_1, U_1, R_1, ALP_1), \dots, (G_n, U_n, R_n, ALP_n)$$

are G and ALP , respectively, such that:

- if R_n is non-empty then
 - G is obtained from G_n by applying R_n
 - $ALP = ALP_n$
- if U_n is non-empty then
 - G is the updated goal wrt G_n and U_n
 - $ALP = U_n(ALP_n)$.

We will refer to a LIFF derivation whose updates are all empty as a **static LIFF derivation**.

5.3 Successful LIFF derivations and extracted answers

Here we formalise the notion of answer extraction for LIFF, by adapting the notion of answer extraction for IFF to take labels into account.

Given a LIFF derivation for a query Q , ALP $\langle P, \mathcal{A}, I \rangle$ and a given sequence of updates, let G be the end goal of the derivation. Let D be a disjunct of G :

- if no inference rule can be applied to D , then D is called **conclusive**;
- if $false : \perp$ is a conjunct in D , then D is called **failed**;
- if D is conclusive and not failed then it is called **successful**.

Then, the derivation is **successful** iff there exists a successful disjunct D in G . An **answer extracted from** a successful LIFF-derivation wrt a query Q , an ALP and a sequence of updates is the set of all abducible atoms, without labels, in a successful disjunct D in the end goal of the derivation.

6 SOUNDNESS OF THE LIFF PROOF PROCEDURE

Theorem 1 (Soundness) *Let E be an answer extracted from a successful LIFF-derivation wrt Q , $\langle P, \mathcal{A}, I \rangle$ and a sequence of updates. Let $\langle P', \mathcal{A}', I' \rangle$ be the ALP resulting after the sequence of updates. Then E is an explanation for Q , wrt $\langle P', \mathcal{A}', I' \rangle$.*

The proof of this theorem relies upon the following lemmas and the correctness of the IFF proof procedure.

Lemma 1 *Every non-static LIFF derivation wrt Q , $\langle P, \mathcal{A}, I \rangle$, U_1, \dots, U_m , with end goal G and end ALP $\langle P', \mathcal{A}', I' \rangle$, can be mapped onto a static LIFF derivation wrt Q , $\langle P', \mathcal{A}', I' \rangle$, ending at (a simplified version of) G .*

Lemma 2 *Every static LIFF derivation wrt Q , $\langle P, \mathcal{A}, I \rangle$, can be mapped onto an IFF derivation wrt Q .*

7 CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a dynamic abductive logic programming proof procedure, called LIFF, which modifies the existing proof procedure IFF by adding labels to goals. The labels keep track of dependencies amongst the atoms and implications in goals and between these and the logic program. By keeping track of these dependencies, after updating the ALP, LIFF can keep parts of the reasoning that have not been affected by the updates and determine how best to replace those parts that have been affected. It thus allows reasoning in dynamic environments and contexts without having to discard earlier reasoning when changes occur. We have considered updates consisting of addition and deletion of if-rules and integrity constraints.

LIFF is particularly good at saving earlier reasoning effort when the definitions of relatively lower level atoms are modified. This kind of update is particularly prominent in the case when we use abductive logic programming and agent-based techniques for information integration [19]. In such an application the higher level atoms are user-level and auxiliary atoms, and the lower level atoms are related to information sources. Changes in various aspects of the information sources, for example content or availability, amount to updating these lower level atoms.

Our work on LIFF shares the same objectives as that of [4]. However whereas we adapt IFF for abductive logic programming, they adapt conventional SLDNF for ordinary logic programs. We also

share some of the objectives of [15, 16]. In their work on speculative computation they allow reasoning to proceed with default values for specific facts and then accommodate updates which replace the default values, attempting to keep some of the earlier computation. They, however, use SLD derivation for Horn theories (no negation in rules and no integrity constraints). They also keep a form of dependency, but it is at a much coarser level compared to ours, as the dependency information is for the whole goal, rather than for the individual conjuncts in goals, thus allowing fewer savings in computations.

We have described LIFF and a soundness result for propositional abductive logic programs. Work is currently in progress for the predicate case, catering for the additional inference rules of factoring, case analysis and equality rewriting. Completeness results are subject of future work. It would also be worth exploring how this work scales up to substantial applications that would require large knowledge bases and frequent updates.

REFERENCES

- [1] C. E. Alchourron, P. Gardenfors, and D. Makinson, ‘On the logic of theory change: Partial meet functions for contraction and revision’, *Journal of Symbolic Logic*, **50**, 510–530, (1985).
- [2] K. L. Clark, ‘Negation as Failure’, in *Logic and Data Bases*, 293–322, Plenum Press, (1978).
- [3] T. H. Fung and R. A. Kowalski, ‘The IFF proof procedure for abductive logic programming’, *Journal of Logic Programming*, **33**(2), 151–165, (1997).
- [4] H. Hayashi, ‘Replanning in robotics by dynamic SLDNF’, in *Proc. Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World*, (1999).
- [5] H. Hayashi, K. Cho, and A. Ohsuga, ‘Integrating planning, action execution, knowledge updates and plan modifications via logic programming’, in *Proc. ICLP*, volume 2401 of *LNCS*. Springer-Verlag, (2002).
- [6] A. C. Kakas, R. A. Kowalski, and F. Toni, ‘The role of abduction in logic programming’, in *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, 235–324, Oxford University Press, (1998).
- [7] A. C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni, ‘The KGP model of agency’, in *Proc. ECAI*, pp. 33–37. IOS Press, (2004).
- [8] R. A. Kowalski and F. Sadri, ‘From logic programming towards multi-agent systems’, *Annals of Mathematics and Artificial Intelligence*, **25**(3/4), 391–419, (1999).
- [9] K. Kunen, ‘Negation in logic programming’, in *Journal of Logic Programming*, volume 4, pp. 289–308, (1987).
- [10] P. Mancarella, F. Sadri, G. Terreni, and F. Toni, ‘Planning partially for situated agents’, in *CLIMA V*, pp. 230–248, (2004).
- [11] F. Sadri and F. Toni, ‘Abduction with negation as failure for active and reactive rules’, in *Proc. AI*IA*, number 1792 in *LNAI*, pp. 49–60. Springer-Verlag, (2000).
- [12] F. Sadri and F. Toni, ‘Interleaving belief revision and reasoning: preliminary report’, in *Proc. CILC*, (2005).
- [13] F. Sadri, F. Toni, and P. Torroni, ‘An abductive logic programming architecture for negotiating agents’, in *Proc. JELIA*, volume 2424 of *LNCS*, pp. 419–431. Springer-Verlag, (2002).
- [14] F. Sadri, F. Toni, and P. Torroni, ‘Dialogues for negotiation: agent varieties and dialogue sequences’, in *Proc. ATAL*, volume 2333 of *LNAI*, pp. 405–421. Springer-Verlag, (2002).
- [15] K. Satoh, K. Inoue, K. Iwanuma, and C. Sakama, ‘Speculative computation by abduction under incomplete communication environments’, in *Proc. ICMAS*, pp. 263–270, (2000).
- [16] K. Satoh and K. Yamamoto, ‘Speculative computation with multi-agent belief revision’, in *Proc. AAMAS*, pp. 897–904, (2002).
- [17] F. Toni, ‘Automated information management via abductive logic agents’, *Journal of Telematics and Informatics*, **18**(1), 89–104, (2001).
- [18] F. Toni and K. Stathis, ‘Access-as-you-need: a computational logic framework for flexible resource access in artificial societies’, in *Proc. ESWA*, *LNAI*, pp. 126–140. Springer-Verlag, (2002).
- [19] I. Xanthakos, *Semantic Integration of Information by Abduction*, Ph.D. dissertation, Imperial College London, 2003.

Bridging the Gap between Informal and Formal Guideline Representations

Andreas Seyfang,¹ Silvia Miksch,^{1,2} Mar Marcos,³
Jolanda Wittenberg,⁴ Cristina Polo-Conde,³ Kitty Rosenbrand⁴

Abstract. Clinical guidelines are important means to improve quality of health care while limiting cost and supporting the medical staff. They are written as free text with tables and figures. Transforming them into a formal, computer-processable representation is a difficult task requiring both computer scientist skills and medical knowledge.

To bridge this gap, we designed an intermediate representation (or ontology) which serves as a mediator between the original text and different formal guideline representations. It is easier to use than the latter, structures the original prose and helps to spot missing information and contradictions.

In this paper we describe the representation and a practical evaluation thereof through the modelling of a real-world clinical guideline.

1 Introduction

Clinical guidelines are "systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances" [2]. A guideline describes the optimal care for patients and therefore, when properly applied, it is assumed that they improve the quality of care. Evidence-based guidelines are becoming an important and indispensable part of quality health care.

Translating guidelines into a computer-processable form brings several advantages. It makes them more accessible to browsing as in DeGeL [8], it allows their execution, i.e., the selection of the appropriate treatment steps based on the patient condition, and it is a precondition to various quality assurance techniques such as those pursued by the Protocure project [1].

While producing a formal model of a guideline is desirable, it is difficult and expensive. In addition, the resulting model is often difficult to compare to the original. If a guideline is revised, the modeling effort is lost and it is not easy to detect which changes in the formal model are required by the changes in the original text.

The main reason for this is that there is a large gap between natural language and the currently available formal representations. To close this gap in a versatile manner we designed an *intermediate representation* called MHB (Many-Headed Bridge). It can be seen as a small and versatile *ontology* of guideline components. It groups the statements in the guideline into chunks with predefined dimensions such as control flow, data flow, temporal aspects and evidence. The aspects of each dimension are described using natural language. Af-

ter building the initial model in MHB, the content of the aspects is reviewed to eliminate deliberate use of synonyms and to describe the knowledge missing to arrive at a model.

As the name suggests, MHB is not designed as a unidirectional bridge from a certain type of natural language guideline to a certain formal representation. Instead it is designed as a versatile device to improve guideline quality. Modeling a guideline in MHB makes important aspects such as control and data flow, resources, and patient aspects explicit. They can easily be grouped in various overview lists. Using the MHB model helps in locating and acquiring knowledge which is missing in the guideline text and inconsistencies such as contradicting definitions or recommendations.

In the next section we describe related work in the field of guideline modelling. In Section 3 we describe MHB in detail. In Section 4 we describe its evaluation. Section 5 concludes the paper.

2 Related Work

There are several guideline representations. Asbru, EON, GLIF, Guide, Prodigy and ProForma have been compared by [4]. Further representation are Glare [11] and GEM [9]. Although the degree of formalization varies between these approaches, the majority of them represents the guideline in a format which is precise enough to execute it (semi-)automatically. In practice, it is very difficult and error-prone to translate the original guideline into such a precise format in one step. Except for GEM and NewGuide, the mentioned languages do not explicitly model the evidence level of the each part of the guideline.

The tool which is used to generate and maintain the guideline's version in MHB forms an important background for the design of the representation itself. This tool is called Document Exploration and Linking Tools with Add-ons (DELT/A) [12]. DELT/A allows for explicit linking between pairs of guideline parts in textual and formal representations. DELT/A supports a multi-step modeling process. E.g., in a first step, the original guideline text is displayed at the left-hand side and the MHB model at the right-hand side. In a second step, the (then complete) MHB model is displayed at the left and a newly created translation of MHB to Asbru, GLIF, or ProForma is shown at the right-hand side. Tools similar to DELT/A are GEM Cutter [9], Degel [8], and Stepper [10]. DELT/A differs from GEM Cutter and Degel in maintaining explicit links between the original text and the formal representation. In contrast to Stepper it does not prescribe a fixed number of modeling steps from the informal text to the formal model.

¹ Institute of Software Technology, Vienna University of Technology, Austria

² Department of Information & Knowledge Engineering, Danube-University Krems, Austria

³ Universitat Jaume I, Spain

⁴ Dutch Institute for Healthcare Improvement, the Netherlands

3 Our Solution: A Many-Headed Bridge

This section describes our solution to the afore described challenges in detail. Several teams try to bridge the gap between different kinds of representation building bridges between pairs of isles. Our vision is to connect all these isles by a single bridge having more than two heads. We therefore call it the *many-headed bridge* (MHB).

The syntax of MHB is based on XML, which allows us to use a range of available tools to create and process it, most importantly DELTA/A.

The overall structure of an MHB file is very flexible. It is a series of *chunks*. Each chunk corresponds to a certain bit of information in the natural language guideline text, e.g., a sentence, part of a sentence, or more than one sentence. Initially, the order of the chunks reflects the order of building blocks in the original version of the guideline. However, they can be moved freely in the MHB file. Such regrouping eases the construction of a more formal representation based on the MHB model.

The information in a chunk is structured in *aspects*. These aspects are grouped in *dimensions*, e.g., control flow, data flow, and evidence. The aspects of different dimensions are independent, i.e., a chunk can have any combination of aspects. However, some are related, e.g., the evidence aspects always refer to the aspects in the other dimensions of the same chunk. In practice, for most chunks only aspects of a few dimensions are given. The reason for this lies in the fact that chunks are a uniform representation for all the heterogeneous parts of the guideline.

In addition to the aspects grouped in dimensions, each chunk has optional fields for the context and the subject of the chunk. All aspects contain link elements which store the connection between each part of the MHB file and the original guideline text. In addition, most aspects contain an attribute *degree-of-certainty* to explicitly show the confidence the original guideline authors expressed in the particular statement, expressed in words like "should", "could", "is advisable", etc. This is completely independent of the evidence dimensions (Section 3.4) which describes the formal rating of the evidence base of a certain conclusion or the importance of a certain recommendation. The subsections below describe the representation of each of these dimensions.

3.1 Dimension: Control Flow

One of the most prominent aspects of a guideline is, *when to do what*. This is often shown (incompletely) by a clinical algorithm. *When* either relates to the condition under which something happens, or to the temporal order of actions relative to each other. *What* generally refers to a task⁵ to be performed under the described condition. A third issue is *decomposition*, i.e., the way in which larger tasks or activities are made up of smaller ones. Furthermore, some tasks are performed *repeatedly* as part of other tasks.

Ordering, decomposition, and repetition imply temporal aspects, e.g., one task being performed after the other. Still, they are included here, while the dimension *temporal aspects* deals with explicit and often quantitative descriptions of the timing of actions and also effects, etc. (see explanations in Section 3.3).

Decisions. From the formal modeling point of view, the main distinction is made between mutually exclusive and non-excluding options. In the first case, clear directives on how to arrive at exactly one

of the given options are specified, be it in a decision tree, a decision table, or simple if-then rules. In the second case, for each of the options there are one or more reasons to take it or not to take it and there is no a-priori safety that exactly one option will be taken.

In MHB the basic structure of a single decision is *if-then*. It consists of a condition, a condition-modifier, a result, a result-modifier, and a degree-of-certainty. All five are attributes formally containing any text. Semantically, the condition is described as precisely as possible based on the guideline text. It should – if possible – contain concepts described in the data dimension (compare Section 3.2). The condition-modifier supports various categories of choices, such as negative recommendations and strong and weak arguments pro and contra a certain option.

The result designates the recommended action, first in the words of the original guideline, later in terms of the more formal representation to which MHB will be translated. If omitted, result-modifier defaults to simply *do* but it can also take values describing the state of the task/plan/action. These depend on the target representation and on the precision of the information in the guideline. E.g., tasks in ProForma have three states (abandoned, terminated, performed) while plans in Asbru only have two (aborted, completed).

The words *should*, *can*, etc. which sometimes are found in a guideline's recommendations as well as phrases like "it seems appropriate" are stored in *degree-of-certainty* alternatively to the above.

Sometimes more than one recommendation or option are described together. In MHB the element *option-group* is used to group several *if-then* elements. The options can exclude each other or not. The attribute *selection-type* has the values *single-choice* or *multiple-choice* to represent this distinction. The additional (optional) attribute *other-selection-specification* can contain more complex constraints on the selection, such as the number of options to select together, e.g., "perform two of the following five tasks".

Ordering and Decomposition. A task can be *decomposed* into subtasks. *Ordering* defines constraints on their execution relative to each other. Both ordering and decomposition are modeled by the same element in MHB named *decomposition*.

It names a task as an attribute and the names of subtasks in separate elements.

The ordering of the children is specified in attribute *ordering* of element *decomposition*. Its values should map to plan orderings available in the targeted formal representation. Binary temporal relations between child tasks are modelled in the temporal dimension (Section 3.3).

Synchronization. When several tasks are performed in parallel or otherwise independent from each other, the question arises when to pursue the rest of the guideline. Several formal guideline representation allow to define those subtasks (*awaited-subtasks* in MHB) which must be completed before the next step is taken in a logical expression, or the number of completed subtasks can be given alternatively.

Repetition. Often one task is performed more than once – either for a certain number of times or at certain times during another task. In any case, there is a task ("envelope task") which lasts during all the repetitions and a subtask ("repeated task") which is performed repeatedly.

⁵ The word *task* is used in this text as a synonym for action, plan, activity, procedure, etc. used by different guideline modelling groups.

Atomic Actions. Some activities or tasks are only described in a sentence without further detailing it. In these cases, they are generally called *atomic actions* which may be too strict if taken literally. Therefore, the MHB element to model such cases is called *clinical-activity*.

3.2 Dimension: Data Flow

Interwoven with control flow is the description of the data processing involved in the diagnosis and treatment of the patient. While the processing of data seems of little importance in the treatment of many diseases, it is often prominently described in the diagnosis part of a guideline. As control flow describes the gathering of information, data flow describes how one piece of information is abstracted from other ones. In MHB, we distinguish the following:

- The definition of a data item is rarely found in the guideline in explicit form. Still, it is necessary for the formal version of the guideline. It consists of a name, a type, and often a range of plausible values and a preferred unit.
- The usage of a data item is made explicit to varying degrees in actions described in the guideline and calculation of other values.
- The input of a data item is sometimes explicitly described in the description of the patient interview or diagnosis.
- abstraction-rules describe the calculation or abstraction of one data item based on others. It is found mostly in descriptions of diagnosis. The time at which the calculation or abstraction is performed may be explicitly stated or not. In the first case, the statement in question has a data flow aspect and a control flow aspect at the same time. In the second case, abstraction is assumed to take place automatically whenever necessary.

3.3 Dimension: Temporal Aspects

Both data and control flow may have temporal aspects. They can be qualitative or quantitative. MHB covers the complexity of Asbru (which has the most complex means to model the temporal dimension) in modeling temporal aspects, but adds more standard concepts such as average or precise duration. For each of start, end, and duration, the minimum, maximum, estimate, and precise value can be given. Of course the precise value excludes others, but the other three values can be combined, i.e., minimum, maximum, and estimate can be given together, if ever found in a guideline. The difference between estimate and precise value lies in the semantic given in the guideline. If start or end are given relative to a certain starting point and it is not obviously the start of the task described, then the reference point must be noted together with the offset in the respective attribute.

In addition to the above, the temporal dimension also models qualitative temporal relations such as "A is started after the end of B". While this could be implemented using the above elements, we provide a distinct element for qualitative relations to improve comprehensibility of the model.

3.4 Dimension: Evidence

An evidence-based guideline builds a bridge from carefully examined pieces of evidence which are obtained for the problem to generally applicable recommendations. While it might be an interesting task to document the foundation of each sentence in the guideline, it will be too tiresome in practice. However, it can be useful for certain parts of the guideline.

Evidence for a statement can appear in explicit and implicit form. For *explicit* references with defined format (summary statement of the evidence, literature references) MHB provides the attributes *grade*, *level*, *importance* and *literature-reference*. They are all optional and can be combined as suitable.

Implicit references in the guideline can be made explicit in the MHB file to help improve the quality of the guideline. To this end, the attribute *is-based-on* contains a reference to another MHB element.

3.5 Dimension: Background Information

Background information describes various aspects of the topic. Their potential to be formally encoded largely varies – some information is aimed at motivating the reader to follow the guideline, while other information complements the statements in the recommendation part.

- *Intentions* of the described actions or recommendations inform and motivate the reader about the reason of certain steps.
- *Effects* are relations between data or phenomena and other phenomena which are not seen as events or actions.
- *Relations* are similar to effects, but do not postulate that one of the two named entities is the cause of the other.
- Other *educational information* can target at the physician or the patient to discuss the recommendations in detail. To some extent, it is related to evidence, as it is an alternative backing for the recommendations in the guideline.
- *Explanations* contain information directly explaining recommendations or other (important) statements in the guideline. When executing a guideline in a computer-based form, it might be useful to show them to the user (patient or physician) – in contrast to the education information above which is not as directly related to the actions resulting from following the guideline.
- *Indicators* are measurable elements of health care that give an indication about the quality. One way to develop indicators is to derive them from key recommendations of a guideline. They are not yet part of many guidelines but will be integrated in the future.

3.6 Dimension: Resources

Each action consumes resources of various nature: *Personnel* resources such as the working time of clinical staff; *Devices*, such as treatment facilities; and *Financial cost*, which can be given independent of qualitative or quantitative information on the above items. It includes also drug cost, etc.

3.7 Dimension: Patient Related Aspects

While the resources dimension mostly represents the view of the care provider, there are several other general issues mentioned in a guideline which see treatment from the patient perspective. *Risk* connected to a certain treatment option or diagnostic action. *Patient discomfort* is often described as free text and refers to undesired side effects of treatments, such as chemotherapy against cancer. *Health prospective* can be seen as a resource which is gained by the treatment process, while it is consumed by waiting.

3.8 Dimension: Document Structure

While the position of a statement in the guideline document could be considered a formal property, its status (narrative, displayed recommendation) certainly forms an important context for its interpretation. This means that a statement found in the recommendations or

control	decomp- osition	parent-task	mastectomy
		child-task(s)	deciding whether to carry out primary or secondary reconstruction
	if-then	condition	higher risk of requiring postoperative radiotherapy
		result	consider the increased risk of complications
	degree-of- certainty	should	
data	usage	name	is there higher risk of postoperative requiring radiotherapy
structure		status	recommendations

Figure 1. Model of a vague recommendation.

the scientific conclusions has more weight than one in the introduction or other parts of the guideline.

Example. Figure 1 shows the MHB model of the statement "*In women who have a higher risk of requiring postoperative radiotherapy, the increased risk of complications should be considered when deciding whether to carry out primary or secondary reconstruction.*" This statement is made in the context of mastectomy interventions. It can be modelled as a decomposition stating that "deciding whether to carry out primary or secondary reconstruction" is a child-task of task "mastectomy". This child-task is performed under the condition of "higher risk of requiring postoperative radiotherapy". There is one data item used here – the information whether there is a higher risk of requiring postoperative radiotherapy.

MHB is intended for the use with various guideline representations. In the first practical evaluation, carried out as part of the Protocure II project, it was used together with Asbru, bridging the gap between this representation and the English text in a real-world guideline of more than 100 pages. This is described in the next section.

4 Evaluation

Evaluation of MHB is performed on a theoretical and on a practical level.

Theoretical evaluation. This consisted of three parts complementing each other:

- The mapping between MHB and various formal representations for clinical guidelines and protocols is analyzed in detail in [7].
- The suitability of MHB to represent guidelines was discussed with domain experts.
- It was shown that MHB can represent the guideline constructs on a list of prototypical patterns developed as another part of the Protocure II project [6].

Each of the three activities led to the conclusion that MHB is sufficiently expressive. The discussion with domain experts also showed that MHB is easier to comprehend than Asbru or other formal guideline representations.

Dimension	Aspect	Chapter						Total
		1	2	3	4	5	6	
control	clinical-activity	13	1	1	1	0	9	25
	decomp.	32	2	4	2	4	4	48
	if-then	39	0	2	2	5	12	60
	option-group	1	6	1	1	1	2	12
	repetition	3	2	1	6	0	0	12
	<i>subtotal</i>	88	11	9	12	10	27	157
data	abstraction	1	0	3	0	2	0	6
	definition	4	8	1	0	2	0	15
	usage	52	31	2	3	14	22	124
	<i>subtotal</i>	57	39	6	3	18	22	145
time	end	1	0	0	0	0	0	1
	qualitative-rel.	4	0	2	1	0	0	7
	start	1	3	0	0	0	0	4
	<i>subtotal</i>	6	3	2	1	0	0	12
evidence	grade	28	5	9	11	7	9	69
	level	37	5	9	11	8	13	83
	literature-ref.	43	4	9	11	8	13	88
	<i>subtotal</i>	108	14	27	33	23	35	240
back- ground	educational	17	7	4	9	2	6	45
	effect	10	8	10	6	3	4	41
	explanation	2	1	1	1	2	0	7
	intention	3	1	2	2	2	3	13
	relation	1	5	6	0	0	0	12
	<i>subtotal</i>	33	22	23	18	9	13	118
patient- aspects	discomfort	1						1
	health-impr.	7						7
	<i>subtotal</i>	10						10
resources	personal-needed	1	2	1				4
	req.-devices		1					1
	<i>subtotal</i>	1	3	1				5
	Total	311	92	67	68	60	97	695

Figure 2. Usage count for each dimension and aspect.

Practical evaluation. Scientific conclusions, other considerations, and recommendations in the Dutch Guideline for the Treatment of Breast Carcinoma [3] were modeled in MHB in a shared effort by three teams in different countries [5]. Parts of the guideline not relevant for the subsequent formal representation of the guideline were omitted. These parts mostly described background information regarding the disease and the details of the studies which form the evidence for the recommendations. All recommendations were modeled.

Table 2 shows the number of elements used for each of the aspects in the various dimensions.

The greatest count of elements is given for dimension evidence which was expected for an evidence-based guideline such as the modeled one. As assumed from the beginning, the control dimension proved to be the most prominent one. The data dimension is also frequently used.

In the background dimensions, educational information and description of effects were equally frequent while all other aspects of that dimension were rarer.

Some rare aspects were not used at all in the guideline. They are omitted for space consideration, as is the structure dimension which

is trivially present in every chunk.

The evaluation of the data dimensions confirms the assertion that data input is only implied in the guideline. There was not a single bit of information found for any chunk. This is in part due to the fact that the guideline deals with treatment and not with diagnose or screening. Besides this, it is assumed that the reader of the guideline knows about the usual data flow during diagnosis and treatment.

The fact that little information regarding patient aspects and resources were found is in line with the common opinion that future guidelines should focus more on these issues. The fact that they were mostly found in chapter 1 hints at an inter-modeler deviation.

Indicators were not contained in the guideline because they are a future development of guidelines.

Overall, the evaluation confirmed the suitability of the representation. All the necessary information was captured in the modelling process.

Unfortunately, the nature of the chapters proved to be rather dissimilar. Therefore, deviations between chapters or teams can be caused by the nature of the chapter or by the preference of the team. However, when reading chapters from other teams, we found a very large degree of consistency of the modeling. This was also due by a set of written recommendations which directed the modeling process.

5 Conclusions

Our experience has shown that MHB is appropriate to model the statements found in the significant guideline parts. MHB not only provides constructs to express the essential knowledge we intended to model, but also allowed for a modeling with the degree of detail necessary for our purposes. An initial problem was the variation observed across the MHB models obtained initially. To solve this, we elaborated a series of basic MHB modeling recommendations. Thanks to these recommendations, the degree of variation was greatly decreased, regardless of the different background of the modellers.

In the Protocure II project we showed that MHB is easier to understand than Asbru by persons without computer background. However, a significant effort in training was necessary.

It is easier to create an MHB model from the original guideline text than an Asbru model. The main reason for this is that MHB does not demand complete information. Also, MHB can be structured like the guideline, while formal representations such as Asbru and others model a guideline as a hierarchy of tasks which is not easy to detect in the original guideline text.

In addition, it is easier to create an Asbru model based on MHB than based on the original text alone. While missing knowledge and vague information in the guideline text still cause modeling problems, they are more efficiently handled since they are already displayed in the MHB model. Using a cross-reference table of all task and data items mentioned in the guideline, we were able to spot countless instances of slightly deviating names for the same entity used in different places of the guideline.

The major drawback of MHB compared to other, more formal representations such as Asbru or GLIF lies in the fact that the syntax of MHB does not impose strict rule for the usage of each attribute (or aspect). The usage is only described in a guidelines [7] and it is the author's responsibility to follow them. While this is an advantage in the early modeling phase, it takes considerable effort to arrive at a uniform naming scheme for tasks and data items in the guideline. However, this is a known problem shared by all formal guideline representations.

In practical work with non-IT persons such as epidemiologists it showed that MHB when used in an XML editor like DELTA is very difficult to understand for them. Therefore, future work will go into developing a user-friendly editing environment, which guides the user by means of context-sensitive help and shields away the complexity of XML. Also, the integration of knowledge extraction to automatically generate parts of the model is an interesting option. Standard vocabularies such as UMLS can be used together with MHB (including compliance checks), however the editor does not yet include browsing and picking items per mouse click.

Weighing the advantages and limitations of MHB, we conclude that MHB is a suitable solution to bridge the gap between the original guideline text and formal representations such as Asbru.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission's IST program, under contract number IST-FP6-508794 Protocure-II.

REFERENCES

- [1] M. Balser, O. Coltell, J. van Croonenborg, C. Duelli, F. van Harmelen, A. Jovell, P. Lucas, M. Marcos, S. Miksch, W. Reif, K. Rosenbrand, A. Seyfang, and A. ten Teije, 'Protocure: Supporting the development of medical protocols through formal methods', in *Computer-based Support for Clinical Guidelines and Protocols*, eds., K. Kaiser, S. Miksch, and S. Tu, pp. 103 – 107. IOS Press, (2004).
- [2] M. J. Field and K. H. Lohr, 'Clinical practice guidelines: Directions for a new program', (1990).
- [3] Nationaal Borstkanker Overleg Nederland (NABON), *Guideline for the treatment of breast carcinoma*, Van Zuiden Communications B.V., 2002.
- [4] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and M. Stefanelli, 'Comparing computer-interpretable guideline models: A case-study approach', *JAMIA*, **10**(1), (2003).
- [5] C. Polo, M. Marcos, A. Seyfang, J. Wittenberg, S. Miksch, and K. Rosenbrand, 'Assessment of MHB: an intermediate language for the representation of medical guidelines', in *Proceedings of CAEPIAO5*, (2005).
- [6] R. Serban, A. ten Teije, F. van Harmelen, M. Marcos, C. Polo, J. C. Galan, P. Lucas, A. Homersom, K. Rosenbrand, J. Wittenberg, and J. van Croonenborg, *Deliverable D2.5: Library of design patterns for guidelines*, EU Project Protocure, 2004. Available at www.protocure.org (Accessed: 1 May 2006).
- [7] A. Seyfang, S. Miksch, P. Votrubka, K. Rosenbrand, J. Wittenberg, J. van Croonenborg, W. Reif, M. Balser, J. Schmitt, T. van der Weide, P. Lucas, and A. Homersom, *Deliverable D2.2a: Specification of Formats of Intermediate, Asbru and KIV Representations*, EU Project Protocure, 2004. Available at www.protocure.org (Accessed: 1 May 2006).
- [8] Y. Shahar, O. Young, E. Shalom, A. Mayaffit, R. Moskovich, A. Hessing, and M. Galperin, 'DeGeL: A hybrid, multiple-ontology framework for specification and retrieval of clinical guidelines', in *Proceedings in Artificial Intelligence in Medicine*, (2003).
- [9] R. N. Shiffman, B. T. Karras, A. Agrawal, R. Chen, L. Marenco, and S. Math, 'GEM: A proposal for a more comprehensive guideline document model using xml', *Journal of the American Medical Informatics Association*, **7**(5), 488–498, (2000).
- [10] V. Svatек and M. Ruzicka, 'Step-by-step mark-up of medical guideline documents', *International Journal of Medical Informatics*, **70**(2-3), 329–335, (2003).
- [11] P. Terenziani, G. Molino, and M. Torchio, 'A modular approach for representing and execution clinical guidelines', *Artificial Intelligence in Medicine*, **23**, 249–276, (2001).
- [12] P. Votrubka, S. Miksch, A. Seyfang, and R. Kosara, 'Tracing the formalization steps of textual guidelines', in *Computer-based Support for Clinical Guidelines and Protocols*, eds., K. Kaiser, S. Miksch, and S. W. Tu, pp. 172–176. IOS Press, (2004).

Boolean Propagation Based on Literals for Quantified Boolean Formulae

Igor Stéphan¹

Abstract. This paper proposes a new set of propagation rules for quantified Boolean formulae based on literals and generated automatically thanks to quantified Boolean formulae certificates. Different decompositions by introduction of existentially quantified variables are discussed in order to construct complete systems. This set of rules is compared with already proposed quantified Boolean propagation rule sets and Stålmarck's method.

1 Introduction

The quantified Boolean formulae validity problem is a generalization of the Boolean formulae satisfiability problem. While the complexity of Boolean satisfiability problem is NP-complete, it is PSPACE-complete for quantified Boolean formulae validity problem. This is the price for more concise representation of many classes of formulae. Many important problems in several search fields have polynomial-time translations to the quantified Boolean formulae validity problem. This is the reason why the implementation of effective tools for deciding the validity of quantified Boolean formulae is an important research issue. Since quantified Boolean formulae may be reduced to (unquantified) Boolean formulae by expansion of the universal quantifiers, the first solution seems to reduce the quantified Boolean formula and then apply a satisfiability algorithm. The main drawback of this approach is that the size of the Boolean propositional formula is in worst-case exponential in the size of the quantified Boolean formula. Most of the recent decision procedures for quantified Boolean formulae validity [23, 22, 17, 13, 19] are extensions of the search-based Davis-Putnam procedure [14] for Boolean satisfiability. Some other decision procedures are based on resolution principle [24] (as Q-resolution [11] which extends the resolution principle for Boolean formulae [15] to quantified Boolean formulae or Quantor [10] which combines efficiently Q-resolution and expansion), quantifier-elimination algorithms [21, 20], or skolemization and SAT solvers [7]. There exists also efficient algorithm for the 2CNF-QBF [3] or useful heuristics for Quantified Renamable Horn Formulas [18], for example.

In [6], a methodology is proposed to construct constraint propagation systems to “constraint satisfaction problems that are based on predefined, explicitly given constraints”. Boolean constraint propagation (see [5] for a small but nice history of it) and Quantified Boolean Constraint Propagation [12] verify this definition. Properties and definitions of these articles are in terms of domain and (Quantified) arc-consistency. Usually constraint propagation systems use implicitly a decomposition by introduction of existentially quantified variables. This decomposition does not allow to capture all the

possible simplifications due to the properties of the connectors in the Boolean lattice. We are more interested by these results in term of propositional logic and logical equivalence as in the Stålmarck's method [25] for tautology checking.

In [8] is introduced the notion of certificate for a QBF. A certificate is a mapping extracted from a QBF which allows to check or generate models. From this certificate we will see in the following that rules for Quantified Boolean Propagation may be extracted automatically.

In this article, after some preliminaries in section 2, we describe in section 3 the core of our contribution: a set of Boolean propagation rules based on literals for QBF. First we introduce the decomposition by introduction of existentially quantified literals ; then we describe the automatic generation, thanks to the certificate, of the Boolean propagation rules based on literals for QBF ; and finally we propose a complete algorithm based on this set of rules and the semantics of the quantifiers. In section 4, we compare our approach with already proposed (quantified) Boolean propagation rule systems and the Stålmarck's method and in section 5, we present some future works.

2 Preliminaries

Quantified boolean formulae. The Boolean values are denoted **true** and **false**. The set of propositional symbols (or variables) is denoted PV . The symbols \perp and \top are the propositional constants. The symbol \wedge is used for conjunction, \vee for disjunction, \neg for negation, \rightarrow for implication and \leftrightarrow for equivalence. A literal is a propositional variable or the negation of a propositional variable. If l is a literal and $l = \neg x$ then $|l| = x$ and $\bar{l} = x$ otherwise $|l| = l$ and $\bar{l} = \neg l$. Propositional satisfaction is denoted \models and logical equivalence is denoted \equiv . The symbol \exists is used for existential quantification and \forall for universal quantification (q is used as a quantification variable). Every Boolean formula is also a quantified Boolean formula (QBF). If F is a QBF and x is a propositional variable then $(\exists x F)$ and $(\forall x F)$ are QBF. If a literal $l = \neg x$ then $q|l|$ stands for qx otherwise it stands for ql . It is assumed that distinct quantifiers bind occurrences of distinct variables. If a variable x is not under the scope of a quantifier qx then it is a free variable. The set of free variables of a QBF F is denoted $FV(F)$. We define $G[x \leftarrow F]$ as the formula obtained from G by replacing occurrences of the propositional variable x by the formula F . A binder Q is a string $q_1x_1 \dots q_nx_n$ with x_1, \dots, x_n distinct variables and $q_1 \dots q_n$ quantifiers. We write $qx_1 \dots x_n$ for any permutation of $qx_1 \dots qx_n$. A QBF QF is in prenex conjunctive normal form if F is a Boolean formula (called the matrix) in conjunctive normal form.

QBF semantics. Semantics of all the Boolean symbols is defined in standard way. In particular, from the structure of the Boolean lat-

¹ LERIA, Université d'Angers, France, email: igor.stephan@info.univ-angers.fr

tice some simplifications on a QBF may be applied (we only present simplifications for disjunction, but a similar presentation may be done for conjunction, implication or any binary Boolean operator):

$$\begin{array}{ll}
 (1) \quad (\perp \vee \perp) \equiv \perp & (2) \quad (\perp \vee \top) \equiv \top \\
 (3) \quad (\top \vee \perp) \equiv \top & (4) \quad (\top \vee \top) \equiv \top \\
 (5) \quad (\perp \vee y) \equiv y & (6) \quad (\top \vee y) \equiv \top \\
 (7) \quad (x \vee \perp) \equiv x & (8) \quad (x \vee \top) \equiv \top \\
 (9) \quad (x \vee x) \equiv x & (10) \quad (x \vee \overline{x}) \equiv \top
 \end{array}$$

These rules may be applied iteratively until the (unique) fix-point is reached. The semantics of QBF is defined as follows: for every Boolean variable y and QBF F , a formula $(\exists y F) = (F[y \leftarrow \top] \vee F[y \leftarrow \perp])$ and $(\forall y F) = (F[y \leftarrow \top] \wedge F[y \leftarrow \perp])$. A QBF F is valid if $F \equiv \top$. If y is an existentially quantified variable preceded by the universally quantified variables x_1, \dots, x_n we denote $\hat{y}_{x_1, \dots, x_n}$ its Skolem function from $\{\text{true}, \text{false}\}^n$ to $\{\text{true}, \text{false}\}$. A model for a QBF F is a sequence s of satisfying Skolem functions for F (denoted $s \models F$). For example, the QBF $\exists y \exists z ((x \vee y) \leftrightarrow z)$ is not valid but the QBF $\forall z \exists y \exists x ((x \vee y) \leftrightarrow z)$ is valid and its possible sequence of satisfying Skolem functions is $\hat{y}_z(v) = v$, $\hat{y}_z(f) = f$, $\hat{x}_z(v) = f$ and $\hat{x}_z(f) = f$. In [27], a new equivalence relation for QBF, denoted \cong is introduced. This equivalence is about preservation of models (and not only preservation of validity). For example, $\forall z \exists y \exists x ((x \vee y) \leftrightarrow z) \equiv \top$ but $\forall z \exists y \exists x ((x \vee y) \leftrightarrow z) \not\cong \top$. A (Boolean) model of an unquantified Boolean formula corresponds exactly to a (QBF) model of its existential closure. A QBF is valid if and only if there exists a sequence of satisfying Skolem functions. According to the theorems $\exists x \exists y F \equiv \exists y \exists x F$, $\forall x \forall y \equiv \forall y \forall x F$ and $\exists x \forall y F \neq \forall y \exists x F$ for any QBF F , the QBF induce an order on the equivalence classes formed by the same adjacent quantifiers. Every QBF may be easily transformed in an equivalent prenex QBF.

Certificates for QBF. In [8] is introduced the notion of certificate for a QBF (this notion is also introduced in [27, 26] but under another name). A certificate is a mapping from the set of the existentially quantified variables of a QBF to couples of Boolean formulae only constituted on the variables which precede the variable in the binder. The certificate may be extracted from a QBF by an extension of the quantifier-elimination algorithm QMRES [20]. From a certificate $\{x \mapsto (\Phi_x^+, \Phi_x^-)\}_{x \in V}$ may be extracted a QBF $\bigwedge_{x \in V} (x \vee \Phi_x^+) \wedge (\neg x \vee \Phi_x^-)$. This QBF is equivalent (i.e. preserves the validity) of the QBF from which the certificate is extracted, but also preserves the set of satisfying Skolem functions (and in other words the models) [27, 26]. This certificate allows model-checking for QBF [8] and also enumeration of models [8, 27, 26]. For example, the mapping $\{y \mapsto (\top, z), x \mapsto ((\neg z \vee y), z)\}$ is the certificate of the QBF $F = \forall z \exists y \exists x ((x \vee y) \leftrightarrow z)$. From this certificate, the QBF $F' = \forall z \exists y \exists x ((y \vee \top) \wedge (\neg y \vee z) \wedge (x \vee (\neg z \vee y)) \wedge (\neg x \vee z))$ may be extracted. This QBF has the two following properties: $F' \equiv F$ and $F' \cong F$.

Decomposition by introduction of existentially quantified variables. The decomposition by introduction of existentially quantified variables (applied usually on prenex formulae) introduces existentially quantified variables to capture the intermediate results of a calculus. This decomposition preserves the validity of the initial QBF. For example, the QBF $\forall z \exists y \exists x \exists u (((x \vee y) \leftrightarrow u) \wedge (\neg z \leftrightarrow v) \wedge ((u \leftrightarrow v) \leftrightarrow \top))$. This QBF

is valid only if the QBF $\forall z \exists v ((\neg z \leftrightarrow v))$, $\exists y \exists x \exists u ((x \vee y) \leftrightarrow u)$ and $\forall z \exists u ((u \leftrightarrow z) \leftrightarrow \top)$ are also valid. This decomposition is the base of Boolean constraint propagation systems [5, 4] and the Stålmarck's algorithm [25] for tautology checking. The simplification rules become simplification propagation rules. In the case of QBF, the binder is important and then added to give what we call an equivalence schema. The simplification rules already introduced are rewritten to integrate the existentially quantified variables and the binder. For example, the simplification rule (9) is rewritten in the equivalence schema $qx \exists z ((x \vee x) \leftrightarrow z)$ with the propagation $[z \leftarrow x]$. Here the order on the quantifiers is very important since $\exists z \forall x ((x \vee x) \leftrightarrow z)$ is not valid. One may notice that rule (10) can not be a rule of Boolean constraint propagation systems based on decomposition by introduction of variables since it does not allow literals in the rules.

3 Boolean Propagation based on literals for Quantified Boolean Formulae

This section describes the core of our contribution: a set of propagation rules for quantified Boolean formulae based on literals and generated thanks to quantified Boolean formulae certificates. First we introduce the decomposition by introduction of existentially quantified literals ; then we describe the automatic generation of the Boolean propagation rules based on literals for QBF ; finally we propose a complete algorithm based on this set of rules and the semantics of quantifiers.

3.1 Decomposition by introduction of existentially quantified literals

The classical decomposition by introduction of existentially quantified variables for Boolean formulae keeps negation as a connector of the generated Boolean formulae. By this way, equivalence schema as $x \vee \overline{x} \equiv \top$ with $[z \leftarrow \top]$ can not be captured. We propose a decomposition based on literals instead of variables to be able to introduce this kind of rules in our propagation system. The negation then disappears form the connectors of the decomposed formula. The following function δ decomposes a Boolean formula by introduction of existentially quantified literals (\circ is a binary connector, functions δ^+ and δ^- return couples (variable, decomposition), $\pi_i(c)$ with $i = 1$ (resp. $i = 2$) stands for the first (resp. second) projection of the couple c).

$$\begin{aligned}
 \delta(F) &= \pi_2(\delta^+(F)) \wedge (\pi_1(\delta^+(F)) \leftrightarrow \top) \\
 \delta^+(x) &= (x, \top), x \in PV \\
 \delta^-(x) &= (\overline{x}, \top), x \in PV \\
 \delta^+(\neg A) &= \delta^-(A), \\
 \delta^-(\neg A) &= \delta^+(A), \\
 \delta^+(A \circ B) &= (z, \pi_2(\delta^+(A)) \wedge \pi_2(\delta^+(B))) \\
 &\quad \wedge ((\pi_1(\delta^+(A)) \circ \pi_1(\delta^+(B))) \leftrightarrow z)) \\
 \delta^-(A \circ B) &= (z, \pi_2(\delta^+(A)) \wedge \pi_2(\delta^+(B))) \\
 &\quad \wedge ((\pi_1(\delta^+(A)) \circ \pi_1(\delta^+(B))) \leftrightarrow \overline{z}))
 \end{aligned}$$

If QF is a prenex QBF, $D = \delta(F)$ the decomposition of F and $X = FV(QD)$ the set of new existentially quantified variables introduce by the function δ then the QBF $Q \exists X D$ is the result of the decomposition by introduction of existentially quantified literals of QF^2 . For example, the QBF $\forall z \exists y \exists x ((x \vee y) \leftrightarrow \neg z)$ is now decomposed in the QBF $\forall z \exists y \exists x \exists u (((x \vee y) \leftrightarrow u) \wedge ((u \leftrightarrow \overline{z}) \leftrightarrow \top))$.

² as in decomposition by introduction of existentially quantified variables the equivalence $z \leftrightarrow \top$ is immediately propagated

According to the theorems $\forall x(F \wedge G) \equiv (\forall xF \wedge \forall xG)$ and $\exists x(F \wedge G) \models (\exists xF \wedge \exists xG)$, if $Q \wedge ((x \circ y) \leftrightarrow z)$ is the decomposition of the QBF F then F is valid only if all the QBF $Q((x \circ y) \leftrightarrow z)$ are also valid.

3.2 The set of Boolean propagation rules based on literals for QBF

The set of Boolean propagation rules based on literals for QBF presented in this article are generated automatically by enumeration of the possible equivalence schemata and by the calculus of the associated certificate as described in the next section³. The set of equivalence schemata is divided in two equal sets: the set of contradictory schemata issued from non valid equivalence schemata and the set of the other schemata. A contradictory rule is generated for every contradictory schema. A contradictory rule must stop the fix-point calculus and return that the QBF is not valid (since one of the conjunction of the decomposed QBF is not valid). The other schemata are of four different types:

- some equivalence schemata are tautological: in that case a tautological simplification rule is generated which only eliminates the equivalence from the decomposition (the number of this rule is superscripted by \models) ;
- some equivalence schemata are only contingent (i.e. it is not tautological and gives no substitution for the variables): in that case no rule is generated (the number of the schema is superscripted by $?$) ;
- some equivalence schemata determine all the variables by substitution: in that case a simplification propagation rule is generated which eliminates the equivalence from the decomposition and propagates the substitutions (the number of this rule is superscripted by \Rightarrow) ;
- some equivalence schemata determine only a part of the variables by substitution and after propagation is still contingent: in that case a propagation rule is generated which propagates the substitutions (the number of this rule is superscripted by $\Rightarrow ?$).

We report only the results for disjunction. We obtain 16 tautological simplification rules (abbreviated in the 10 rules of Figure 2), 58 simplification propagation rules (abbreviated in the 10 rules of the section 2 and the 30 first rules of Figure 1), 2 propagation rules (the last 2 rules of Figure 1), 28 contingent equivalence schemata (cf. Figure 3), and 104 contradictory rules not reported here.

The proof that the fix-point of the iterative application of our rules is always reached and is unique is based on the results of [4]. The argument cannot be based on reduction domain as for constraint propagation systems since not all the rules decrease the number of possible Boolean values of the variables of the rule. Instead we use a usual argument in logic based on the weight of the decomposed formula.

3.3 Automatic generation of the Boolean propagation rules based on literals for QBF

The calculus of a certificate for an equivalence schema (with the literal x (resp. \bar{x}) considers as the variable x (resp. the formula $\neg x$) allows us to deduce automatically its associated rule if exists. During this calculus two cases may appear: the certificate demonstrates that the equivalence is not valid then a contradictory rule is generated or it demonstrates that the equivalence is valid and in that case

	Equivalence schema	Substitutions
(1) \Rightarrow	$\exists y \perp \vee y \leftrightarrow \perp$	$[y \leftarrow \perp]$
(2) \Rightarrow	$\exists y \perp \vee y \leftrightarrow \top$	$[y \leftarrow \top]$
(3) \Rightarrow	$\exists x x \vee \perp \leftrightarrow \perp$	$[x \leftarrow \perp]$
(4) \Rightarrow	$\exists x x \vee \perp \leftrightarrow \top$	$[x \leftarrow \top]$
(5) \Rightarrow	$\exists x x \vee x \leftrightarrow \perp$	$[x \leftarrow \perp]$
(6) \Rightarrow	$\exists x x \vee x \leftrightarrow \top$	$[x \leftarrow \top]$
(7) \Rightarrow	$\exists x x \vee y \leftrightarrow \perp$	$[x \leftarrow \perp]$
(8) \Rightarrow	$\exists x x \vee y \leftrightarrow \top$	$[x \leftarrow \top]$
(9) \Rightarrow	$\exists x y \vee x \leftrightarrow \perp$	$[y \leftarrow \perp]$
(10) \Rightarrow	$\exists x y \vee x \leftrightarrow \top$	$[y \leftarrow \top]$
(11) \Rightarrow	$\exists y \perp \vee y \leftrightarrow \perp$	$[y \leftarrow \perp]$
(12) \Rightarrow	$\exists y \perp \vee y \leftrightarrow \top$	$[y \leftarrow \top]$
(13) \Rightarrow	$\exists x x \vee \perp \leftrightarrow \perp$	$[x \leftarrow \perp]$
(14) \Rightarrow	$\exists x x \vee \perp \leftrightarrow \top$	$[x \leftarrow \top]$
(15) \Rightarrow	$\exists x x \vee x \leftrightarrow \perp$	$[x \leftarrow \perp]$
(16) \Rightarrow	$\exists x x \vee x \leftrightarrow \top$	$[x \leftarrow \top]$
(17) \Rightarrow	$\exists x y x \vee y \leftrightarrow \perp$	$[x \leftarrow \perp], [y \leftarrow \perp]$
(18) \Rightarrow	$\exists x y x \vee y \leftrightarrow \top$	$[x \leftarrow \top]$
(19) \Rightarrow	$\exists y x x \vee y \leftrightarrow \top$	$[y \leftarrow \top]$
(20) \Rightarrow	$\exists x x \vee \top \leftrightarrow x$	$[x \leftarrow \top]$
(21) \Rightarrow	$\exists x x \vee \top \leftrightarrow \bar{x}$	$[x \leftarrow \perp]$
(22) \Rightarrow	$q z \exists y \perp \vee y \leftrightarrow z$	$[y \leftarrow z]$
(23) \Rightarrow	$\exists y \top \vee y \leftrightarrow y$	$[y \leftarrow \top]$
(24) \Rightarrow	$\exists y \top \vee y \leftrightarrow \bar{y}$	$[y \leftarrow \perp]$
(25) \Rightarrow	$\exists z q y \top \vee y \leftrightarrow z$	$[z \leftarrow \top]$
(26) \Rightarrow	$q z \exists x x \vee \perp \leftrightarrow z$	$[z \leftarrow x]$
(27) \Rightarrow	$\exists z q x x \vee \top \leftrightarrow z$	$[z \leftarrow \top]$
(28) \Rightarrow	$q z \exists x x \vee x \leftrightarrow z$	$[x \leftarrow z]$
(29) \Rightarrow	$\exists x x \vee \bar{x} \leftrightarrow x$	$[x \leftarrow \top]$
(30) \Rightarrow	$\exists x x \vee \bar{x} \leftrightarrow \bar{x}$	$[x \leftarrow \perp]$
(31) \Rightarrow	$\exists z q x x \vee \bar{x} \leftrightarrow z$	$[z \leftarrow \top]$
(32) \Rightarrow	$\exists x y x \vee y \leftrightarrow \bar{x}$	$[x \leftarrow \perp][y \leftarrow \top]$
(33) \Rightarrow	$\exists x y x \vee y \leftrightarrow x$	$[x \leftarrow \top]$
(34) \Rightarrow	$\exists y v x x \vee y \leftrightarrow x$	$[y \leftarrow \top]$
(35) \Rightarrow	$\exists x v y x \vee y \leftrightarrow y$	$[x \leftarrow \perp]$
(36) \Rightarrow	$\exists y v x x \vee y \leftrightarrow y$	$[y \leftarrow \top]$
(37) \Rightarrow	$\exists x y x \vee y \leftrightarrow \bar{y}$	$[x \leftarrow \top][y \leftarrow \perp]$
(38) \Rightarrow	$\exists y z v x x \vee y \leftrightarrow z$	$[y \leftarrow \top][z \leftarrow \top]$
(39) \Rightarrow	$\exists y v z \exists x x \vee y \leftrightarrow z$	$[y \leftarrow \top][x \leftarrow z]$
(40) \Rightarrow	$\exists x z v y x \vee y \leftrightarrow z$	$[x \leftarrow \top][z \leftarrow \top]$
(41) \Rightarrow	$\exists x v z \exists y x \vee y \leftrightarrow z$	$[x \leftarrow \perp][y \leftarrow z]$
(1) $\Rightarrow ?$	$\exists z v x \exists y x \vee y \leftrightarrow z$	$[z \leftarrow \top]$
(2) $\Rightarrow ?$	$\exists z v y \exists x x \vee y \leftrightarrow z$	$[z \leftarrow \top]$

Figure 1. Simplification propagation and (only) propagation rules for disjunction.

	Equivalence schema		Equivalence schema
(1) \models	$\perp \vee \perp \leftrightarrow \perp$	(2) \models	$\perp \vee \top \leftrightarrow \top$
(3) \models	$\top \vee \perp \leftrightarrow \top$	(4) \models	$\top \vee \top \leftrightarrow \top$
(5) \models	$q x x \vee \perp \leftrightarrow \top$	(6) \models	$q x x \vee \bar{x} \leftrightarrow \top$
(7) \models	$q y \top \vee y \leftrightarrow \top$	(8) \models	$q x x \vee x \leftrightarrow x$
(9) \models	$q y \perp \vee y \leftrightarrow y$	(10) \models	$q x x \vee \perp \leftrightarrow \perp$

Figure 2. Tautological simplification rules for disjunction.

³ The Prolog program is available on our web site: <http://www.info.univ-angers.fr/pub/stephan/Research/QBF/index.html>

	Equivalence schema		Equivalence schema
(1)?	$\exists[x][y]\exists[z] x \vee y \leftrightarrow z$	(2)?	$\exists[x]\forall[y]\exists[z] x \vee y \leftrightarrow z$
(3)?	$\forall[x]\exists[y]\exists[z] x \vee y \leftrightarrow z$	(4)?	$\exists[y]\forall[x]\exists[z] x \vee y \leftrightarrow z$
(5)?	$\forall[y]\exists[x]\exists[z] x \vee y \leftrightarrow z$	(6)?	$\forall[x][y]\exists[z] x \vee y \leftrightarrow z$
(7)?	$\exists[x][y] x \vee y \leftrightarrow \top$	(8)?	$\forall[x]\exists[y] x \vee y \leftrightarrow \top$
(9)?	$\forall[y]\exists[x] x \vee y \leftrightarrow \top$	(10)?	$q[x]\exists[y] x \vee y \leftrightarrow x$
(11)?	$q[y]\exists[x] x \vee y \leftrightarrow x$	(12)?	$\exists[x][y] x \vee y \leftrightarrow y$
(13)?	$\forall[x]\exists[y] x \vee y \leftrightarrow y$	(14)?	$\forall[y]\exists[x] x \vee y \leftrightarrow y$
(15)?	$q[x]\exists[z]\exists[y] x \vee y \leftrightarrow z$	(16)?	$q[y]\exists[z]\exists[x] x \vee y \leftrightarrow z$
(17)?	$\exists[z]\exists[x][y] x \vee y \leftrightarrow z$	(18)?	$\forall[z]\exists[x][y] x \vee y \leftrightarrow z$

Figure 3. Contingent equivalence schemata for disjunction.

the certificate itself allows us to deduce if the rule is a tautological simplification rule, a simplification propagation rule, a propagation rule or that the equivalence is contingent.

- If the certificate is empty or $\{x \mapsto (\top, \top)\}$ then the equivalence is tautological and a tautological simplification rule is generated.
- If the certificate contains the couple $(x \mapsto (\top, \perp))$ then x is equivalent to \perp and a propagation rule which contains the substitution $[x \leftarrow \perp]$ is generated. Conversely, If the certificate contains the couple $(x \mapsto (\perp, \top))$ then x is equivalent to \top and a propagation rule which contains the substitution $[x \leftarrow \top]$ is generated.
- If the certificate contains the couple $(x \mapsto (\overline{y}, y))$ and if $x < y$ then a propagation rule which contains the substitution $[y \leftarrow x]$ is generated otherwise a propagation rule which contains the substitution $[x \leftarrow y]$ is generated. Conversely, if the certificate contains the couple $(x \mapsto (y, \overline{y}))$ and if $x < y$ then a propagation rule which contains the substitution $[y \leftarrow \overline{x}]$ is generated otherwise a propagation rule which contains the substitution $[x \leftarrow \overline{y}]$ is generated.

If all the literals of an equivalence are determined by substitution then the rule is a simplification rule (which can also be a propagation rule). A propagation rule is not necessarily a simplification rule since the equivalence schema can propagate a Boolean value for a literal and be at the same time contingent.

For example, $\{x \mapsto (\top, \top)\}$ is the certificate of the equivalence $\exists x(x \vee \top \leftrightarrow \top)$ and then is a tautological schema and generates a tautological simplification rule ; $\{y \mapsto (\top, \top), z \mapsto (\neg x \wedge \neg y, x \vee y)\}$ is the certificate of the equivalence $\forall x \exists y \exists z(x \vee y \leftrightarrow z)$ and then is a contingent schema and generates no rule ; $\{x \mapsto (\neg z, z)\}$ is the certificate of the equivalence $\forall z \exists x(x \vee x \leftrightarrow z)$ and then generate a simplification propagation rule ; $\{z \mapsto (\perp, \top), y \mapsto (x, \top)\}$ is the certificate of the equivalence $\exists z \forall x \exists y(x \vee y \leftrightarrow z)$ and then generates a propagation rule with the substitution $[z \leftarrow \top]$ since all the variables are not determined by substitution and the equivalence schema $\forall x \exists y(x \vee y \leftrightarrow \top)$ is still contingent⁴.

3.4 An algorithm to reach completeness

It is obvious that only the application of the set of rules described in this article is incomplete to decide if a QBF is valid or not. In Boolean constraint propagation, completeness is reached by a two-steps loop: propagation-enumeration of one of the remaining variables. Since all

⁴ All the certificates are accessible on our web site.

the variables are existentially quantified, the choice of the variable is free and only guided by efficiency. It can not be the same for QBF since the initial quantifiers are ordered and can not be so easily permuted.

In the case of search algorithms, which try to eliminate the outermost quantifiers first, only the quantifiers of the outermost equivalence class induce by the order may be chosen. Let qxD be the decomposition of a QBF F , x a variable of the outermost equivalence class and $z \in \{\top, \perp\}$. By the semantics of universal quantifier, if $q = \forall$ then $F \equiv (D[x \leftarrow \top] \wedge D[x \leftarrow \perp])$. Then F is valid if and only if $D[x \leftarrow \top]$ and $D[x \leftarrow \perp]$ are valid. In particular, if $D[x \leftarrow z]$ is not valid then it is useless to calculate $D[x \leftarrow \bar{z}]$ and F is not valid. Conversely, by the semantics of existential quantifier, if $q = \exists$ then $F \equiv (D[x \leftarrow \top] \vee D[x \leftarrow \perp])$. Then F is valid if and only if $D[x \leftarrow \top]$ or $D[x \leftarrow \perp]$ are valid. In particular, if $D[x \leftarrow z]$ is valid then it is useless to calculate $D[x \leftarrow \bar{z}]$ and F is valid. We recognize here the Dilemma rule of the Stålmarck's method [25].

It does not seem to us easy to use an inside-out quantifier-elimination method like [20] to reach completeness since inner-most quantifiers are existential and the semantics of existential quantifier introduces a disjunction that breaks the decomposition.

It is worth to notice that if the formula is already decomposed as a conjunction of equivalence schemata all the rules are applicable. Otherwise, the decomposition introduce only one occurrence of each new existentially quantified literals in the right hand side of the equivalences. Then this occurrence can never be universally quantified. This means that some of the rules (mainly contradictory rules) can not be applied.

4 Comparisons

Quantified Boolean propagation. In [12] an extension of arc-consistency to quantified constraints (quantified arc-consistency) is proposed. This extension is applied to QBF and a set of propagation rules is described. This set is the counterpart in constraint propagation of the rules $(1)^{\Rightarrow} - (8)^{\Rightarrow}$, $(11)^{\Rightarrow} - (14)^{\Rightarrow}$, $(17)^{\Rightarrow} - (19)^{\Rightarrow}$, $(22)^{\Rightarrow} - (25)^{\Rightarrow}$, $(38)^{\Rightarrow} - (41)^{\Rightarrow}$, $(1)^{\Rightarrow?}$ and $(2)^{\Rightarrow?}$. The other simplification propagation rules are out of the scope of quantified arc-consistency. So our set of rules is more powerful than the one proposed in [12].

Stålmarck's method. In [25] is described the Stålmarck's method of tautology checking. Tautology checking is co-NP complete and thus corresponds to a prenex QBF with only universally quantified variables. Stålmarck's method tries to prove that the matrix of the QBF can not be equivalent to \perp . Stålmarck's method first uses the same decomposition principal as described in preliminaries and then applies a set of rules. This method translates $(\neg x)$ in $(x \rightarrow \perp)$ and only covers the $\{\rightarrow, \perp\}$ -Boolean formulae. So, initial Stålmarck's rules are for the implication, we translate them (cf. Figure 4) to compare with ours. We do not detail the comparison of the two sets of rules. Just some remarks: rule $(32)^{\Rightarrow}$ covers rule r_6 but is finer since it also substitutes the y variable ; rules $(9)^{\Rightarrow}$, $(16)^{\Rightarrow}$, $(28)^{\Rightarrow}$ and $(37)^{\Rightarrow}$ are not covered by any rules of Stålmarck's method and would be added in a Stålmarck's method based on literals. Stålmarck's method also includes contradictory equivalence schemata (called “terminal triplets”), they are also expressed with implication and given in Figure 4 for comparison. Due to the lack of space we cannot present all our contradictory equivalence schemata. These schemata cover more than the Stålmarck's set of contradictory equivalence schemata and the following ones may be added in a Stål-

marck's method based on literals: $\exists|x| \ x \vee \bar{x} \leftrightarrow \perp$, $\exists|y| \ \perp \vee y \leftrightarrow \bar{y}$, $\exists|x| \ x \vee x \leftrightarrow \bar{x}$ and $\exists|x| \ x \vee \perp \leftrightarrow \bar{x}$.

Stålmarck's set of rules		
	Schema	Substitutions
r_1	$\exists xy \ x \vee y \leftrightarrow \perp$	$[x \leftarrow \perp][y \leftarrow \perp]$
r_2	$\exists xz \ x \vee \top \leftrightarrow z$	$[z \leftarrow \top]$
r_3	$\exists yz \ \top \vee y \leftrightarrow z$	$[z \leftarrow \top]$
r_4	$\exists yz \ \perp \vee y \leftrightarrow z$	$[z \leftarrow y]$
r_5	$\exists xz \ x \vee \perp \leftrightarrow z$	$[z \leftarrow x]$
r_6	$\exists xy \ x \vee y \leftrightarrow \bar{x}$	$[x \leftarrow \top]$
r_7	$\exists xz \ x \vee \bar{x} \leftrightarrow z$	$[z \leftarrow \top]$

Terminal schemata	
	$\perp \vee \perp \leftrightarrow \top$
$\exists y$	$\top \vee y \leftrightarrow \perp$
$\exists x$	$x \vee \top \leftrightarrow \perp$

Figure 4. Stålmarck's set of rules and terminal schemata

Stålmarck's method proves that a propositional formula is a tautology by proving that it is impossible to falsify it. We can not do the same since there exists QBF with no models such that the matrix in prenex normal form has (Boolean) models (for example, the equivalence $\forall x \forall y \forall z (x \vee y \leftrightarrow z)$).

5 Future Works

The RuleMiner algorithm [1] is an algorithm for generating propagation rules for constraints over finite domains defined extensionally. It seems to be able to generate a set of rules more compact and more powerful than the methodology described in [6] thanks to an order over the rules and a more powerful set of possible rules. So we will study the impact of RuleMiner on our set of rules.

We develop a C++ version of a complete algorithm based on our set of rules. This implementation will work on non CNF formulae since during CNF transformation many useful pieces of information are lost. Non CNF benchmarks are rare but in [2] new difficult ones in non CNF format are proposed. One may also use to reduce the scope of quantifiers tools like qTree [9].

The Constraint Handling Rule (CHR) language [16] is used straightforwardly to implement Boolean constraint propagation system. We are interested in implementing our set of rules in CHR in side a dialect of Prolog.

6 Conclusion

In this article we have proposed a new set of Boolean propagation rules based on a decomposition by introduction of existentially quantified literals instead of variables. This new set of rules has been generated automatically thanks to QBF certificate calculated for equivalence schemata. This set of rules has been proven to cover the already proposed set of rules in [12] and to extend the propositional rules of the Stålmarck's method.

REFERENCES

- [1] S. Abdennadher and C. Rigotti, ‘Automatic generation of propagation rules for finite domains’, in *Principles and Practice of Constraint Programming*, pp. 18–34, (2000).
- [2] C. Ansotegui, C. Gomes, and B. Selman, ‘Achilles’ heel of qbf’, in *AAAI*, (2005).
- [3] B. Apsvall, M. Plass, and R. Tarjan, ‘A linear-time algorithm for testing the truth of certain quantified boolean formulas and its evaluation’, *Information Processing Letters*, **8**, 121–123, (1979).
- [4] K.R. Apt, ‘The essence of constraint propagation’, *Theoretical Computer Science*, **221**(1-2), 179–210, (1999).
- [5] K.R. Apt, ‘Some remarks on boolean constraint propagation’, in *New trends in constraints*, volume 1865, Springer-Verlag, (2000).
- [6] K.R. Apt and E. Monfroy, ‘Constraint programming viewed as rule-based programming’, *Theory and Practice of Logic Programming (TPLP)*, **1**(6), 713–750, (2001).
- [7] M. Benedetti, ‘Evaluating QBFs via Symbolic Skolemization’, in *Proc. of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR04)*, number 3452 in LNCS, Springer, (2005).
- [8] M. Benedetti, ‘Extracting Certificates from Quantified Boolean Formulas’, in *Proc. of 9th International Joint Conference on Artificial Intelligence (IJCAI05)*, (2005).
- [9] M. Benedetti, ‘Quantifier Trees for QBFs’, in *Proc. of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT05)*, (2005).
- [10] A. Biere, ‘Resolve and expand’, in *SAT*, (2004).
- [11] H. K. Büning, M. Karpinski, and A. Flögel, ‘Resolution for quantified boolean formulas’, *Information and Computation*, **117**(1), 12–18, (1995).
- [12] L. Bordeaux, ‘Boolean and interval propagation for quantified constraints’, in *First International Workshop on Quantification in Constraint Programming*, (2005).
- [13] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi, ‘An algorithm to evaluate quantified boolean formulae and its experimental evaluation’, *Journal of Automated Reasoning*, **28**(2), 101–142, (2002).
- [14] M. Davis, G. Logemann, and D. Loveland, ‘A machine program for theorem-proving’, *Communication of the ACM*, **5**, (1962).
- [15] M. Davis and H. Putnam, ‘A computing procedure for quantification theory’, *Journal of the ACM*, **7**(3), 201–215, (July 1960).
- [16] T. Fröhwirth, ‘Theory and practice of constraint handling rules’, *Journal of Logic Programming*, **37**(1-3), 95–138, (1998).
- [17] E. Giunchiglia, M. Narizzano, and A. Tacchella, ‘Backjumping for quantified boolean logic satisfiability’, *Artificial Intelligence*, **145**, 99–120, (2003).
- [18] F. Letombe, ‘Une heuristique de branchement dirigée vers les formules horn renommables quantifiées’, in *RJCAI05*, pp. 183–196, (2005).
- [19] R. Letz, ‘Lemma and model caching in decision procedures for quantified boolean formulas’, in *TABLEAUX*, pp. 160–175, (2002).
- [20] G. Pan and M.Y. Vardi, ‘Symbolic decision procedures for qbf’, in *International Conference on Principles and Practice of Constraint Programming*, (2004).
- [21] D.A. Plaisted, A. Biere, and Y. Zhu, ‘A staisfiability procedure for quantified boolean formulae’, *Discrete Applied Mathematics*, **130**, 291–328, (2003).
- [22] J. Rintanen, ‘Improvements to the evaluation of quantified boolean formulae’, in *IJCAI*, pp. 1192–1197, (1999).
- [23] J. Rintanen, ‘Partial implicit unfolding in the davis-putnam procedure for quantified boolean formulae’, in *Workshop on Theory and Applications of QBF. Int. Join Conference on Automated Reasoning*, Sienna, Italia, (2001).
- [24] J.A. Robinson, ‘A machine-oriented logic based on the resolution principle’, *JACM*, **12**(1), 23–41, (1965).
- [25] Mary Sheeran and Gunnar Stålmarck, ‘A tutorial on Stålmarck’s proof procedure for propositional logic’, in *Formal Methods in Computer-Aided Design*, eds., G. Gopalakrishnan and P. Windley, volume 1522, 82–99, Springer-Verlag, Berlin, (1998).
- [26] I. Stéphan, ‘Algorithmes d’élimination de quantificateurs pour le calcul des politiques des formules booléennes quantifiées’, in *Premières Journées Francophones de Programmation par Contraintes*, (2005).
- [27] I. Stéphan, ‘Finding models for quantified boolean formulae’, in *First International Workshop on Quantification in Constraint Programming*, (2005).

General Concept Inclusions in Fuzzy Description Logics

Giorgos Stoislos¹ and **Umberto Straccia²** and **Giorgos Stamou³** and **Jeff Z. Pan⁴**

Abstract. *Fuzzy Description Logics* (fuzzy DLs) have been proposed as a language to describe structured knowledge with vague concepts. A major theoretical and computational limitation so far is the inability to deal with *General Concept Inclusions* (GCIs), which is an important feature of classical DLs. In this paper, we address this issue and develop a calculus for fuzzy DLs with GCIs.

1 INTRODUCTION

Description Logics (DLs) [2] are a logical reconstruction of the so-called frame-based knowledge representation languages, with the aim of providing a simple well-established Tarski-style declarative semantics to capture the meaning of the most popular features of structured representation of knowledge. Nowadays, DLs have gained even more popularity due to their application in the context of the *Semantic Web*, as the theoretical counterpart of OWL DL (the W3C standard for specifying ontologies, see [9] for details).

Fuzzy DLs [15, 18, 23, 24] extend classical DLs by allowing to deal with *fuzzy/vague/imprecise concepts* such as “Candia is a creamy white rose with dark pink edges to the petals”, “Jacaranda is a hot pink rose”, and “Calla is a very large, long white flower on thick stalks”. Such concepts involve so-called *fuzzy or vague concepts*, like “creamy”, “dark”, “hot”, “large” and “thick”, for which a clear and precise definition is not possible.

The problem to deal with imprecise concepts has been addressed several decades ago by Zadeh [25], which gave birth in the meanwhile to the so-called *fuzzy set and fuzzy logic theory* and a huge number of real life applications exists. Despite the popularity of fuzzy set theory, relative little work has been carried out in extending DLs towards the representation of imprecise concepts, notwithstanding DLs can be considered as a quite natural candidate for such an extension [1, 3, 4, 5, 6, 11, 12, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 26].

A major theoretical and computational limitation so far of fuzzy DLs is the inability to deal with *General Concept Inclusions* (GCIs), which is an important feature of classical DLs; e.g., GCIs are necessary to represent domain and range constraints. In this paper, we address this issue and develop a calculus for fuzzy DLs with GCIs.

In the next section, we briefly recall basic concepts of DLs and fuzzy DLs, while in Section 3 we present a sound and complete calculus dealing with GCIs. Section 4 concludes.

2 PRELIMINARIES

DLs basics. DLs [2] are a family of logics for representing structured knowledge. Each logic is identified by a name made of labels,

which identify the operators allowed in that logic. Major DLs are the so-called logic *ALC* [13] and is used as a reference language whenever new concepts are introduced in DLs, *SHOIN(D)*, which is the logic behind the ontology description language OWL DL and *SHIF(D)*, which is the logic behind OWL LITE, a slightly less expressive language than OWL DL (see [9]). A DL can be seen as a restricted First Order Language with unary and binary predicates. For the sake of our purpose we deal here with *ALC*, whose syntax and semantics is described in Table 1, (an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ has domain $\Delta^{\mathcal{I}}$ and maps concepts into subsets of $\Delta^{\mathcal{I}}$, maps roles into subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and maps individuals into elements of $\Delta^{\mathcal{I}}$). An *ALC knowledge base* is defined as a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, where \mathcal{T} is called a *TBox* and \mathcal{A} an *ABox*. \mathcal{T} is a finite set of *general inclusion axioms* (GCIs for short) of the form, $C \sqsubseteq D$ and \mathcal{A} is a finite set of *concept and role assertions* of the form $a : C$ and $(a, b) : R$, respectively. For example \mathcal{T} could contain an axioms of the form *HappyFather* $\sqsubseteq \exists \text{hasChild}.\text{Female}$, and \mathcal{A} an assertion of the form *Tom* : *HappyFather*. An interpretation \mathcal{I} satisfies \mathcal{T} if $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ for all GCIs in \mathcal{T} , then \mathcal{I} is called a *model* of \mathcal{T} , and it satisfies \mathcal{A} if $a^{\mathcal{I}} \in C^{\mathcal{I}} \wedge (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for all concept (role) assertions in \mathcal{A} . Then \mathcal{I} is called a model of \mathcal{A} . An interpretation satisfies an *ALC KB* $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies both \mathcal{A} and \mathcal{T} ; then \mathcal{I} is called a *model* of Σ . A concept C is *subsumed* by a concept D , written $C \sqsubseteq D$, with respect to (w.r.t.) \mathcal{T} , if for all models \mathcal{I} of \mathcal{T} , it holds that $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$. An ABox \mathcal{A} is *consistent* (*inconsistent*) w.r.t. a TBox \mathcal{T} if there exists (does not exist) a model \mathcal{I} of \mathcal{T} , that is also a model of \mathcal{A} . Finally, Σ *entails* an *ALC* assertion ϕ , written $\Sigma \models \phi$, if each model of Σ is a model of ϕ .

Fuzzy DLs basics. Fuzzy DLs [18] extend classical DLs by allowing to deal with *fuzzy/imprecise concepts*. The main idea underlying fuzzy DLs is that an assertion $a:C$, stating that the constant a is an instance of concept C , rather being interpreted as either true or false, will be mapped to a truth value $n \in [0, 1]_{\mathbb{Q}}$ (the rationals in the unit interval $[0, 1]$). The intended meaning is that n indicates to which extent ‘ a is a C ’. From a syntax point of view, concepts, roles, individuals and concept inclusion axioms are as for *ALC*. In place of assertions, we have *fuzzy assertions* [18], which are of the form $\langle \alpha \bowtie n \rangle$, where α is an assertion, $n \in [0, 1]_{\mathbb{Q}}$ and \bowtie is one of $\geq, \leq, >, <$. For instance, $\langle a:C \geq n \rangle$ allows to state that individual a is an instance of concept C at least to degree n . Similarly for role assertions. A *Fuzzy Knowledge Base*, $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, is as for the crisp case, except that now \mathcal{A} is a set of fuzzy assertions rather than assertions only. From a semantics point of view, a *fuzzy interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ has domain $\Delta^{\mathcal{I}}$, but now maps a concept C into a function $C^{\mathcal{I}}: \Delta^{\mathcal{I}} \rightarrow [0, 1]_{\mathbb{Q}}$ and a role R into a function $R^{\mathcal{I}}: \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]_{\mathbb{Q}}$. For $d \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}}(d)$ gives us the degree of d being an instance of the concept C (similarly for roles). The semantics is summarized in Table 1. A fuzzy interpretation \mathcal{I} satisfies a fuzzy TBox \mathcal{T} if $\forall x \in \Delta^{\mathcal{I}}. C^{\mathcal{I}}(x) \leq D^{\mathcal{I}}(x)$ for all

¹ National and Technical University of Athens

² ISTI - Italian National Research Council at Pisa

³ National and Technical University of Athens

⁴ Department of Computing Science, The University of Aberdeen, UK

Table 1. \mathcal{ALC} and fuzzy \mathcal{ALC} .

Syntax		Concepts		Examples
		Classical Semantics	Fuzzy Semantics	
$C, D \rightarrow$	\top (top concept)	$\top^{\mathcal{I}}$	$= \Delta^{\mathcal{I}}$	
	\perp (bottom concept)	$\perp^{\mathcal{I}}$	$= \emptyset$	
	A (atomic concept)	$A^{\mathcal{I}}$	$\subseteq \Delta^{\mathcal{I}}$	
$C \sqcap D$	(concept conjunction)	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \sqcap C_2^{\mathcal{I}}$	$(\bar{C}_1 \sqcup \bar{C}_2)^{\mathcal{I}} = \max(C_1^{\mathcal{I}}, C_2^{\mathcal{I}})$	Human
$C \sqcup D$	(concept disjunction)	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \sqcup C_2^{\mathcal{I}}$	$(\bar{C}_1 \sqcap \bar{C}_2)^{\mathcal{I}} = \min(C_1^{\mathcal{I}}, C_2^{\mathcal{I}})$	Human \sqcap Male
	$\neg C$ (concept negation)	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$(\neg C)^{\mathcal{I}} = \inf_{y \in \Delta^{\mathcal{I}}} \{1 - R^{\mathcal{I}}(x, y)\}$	Nice \sqcup Rich
$\exists R.C$	(existential quantification)	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y, \in \rangle R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y, \in \rangle R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	\neg Male
$\forall R.C$	(universal quantification)	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y, \in \rangle R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y, \in \rangle R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	\exists has.child.Blonde
				\forall has.child.Human

$C \sqsubseteq D \in \mathcal{T}$, then \mathcal{I} is called a *model* of \mathcal{T} , and it satisfies a fuzzy ABox \mathcal{A} if $C^{\mathcal{I}}(a^{\mathcal{I}}) \bowtie n (R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \bowtie n)$, for each $\langle a : C \bowtie n \rangle (\langle a, b : R \bowtie n \rangle)$ in \mathcal{A} . Then \mathcal{I} is called a *model* of \mathcal{A} . A fuzzy \mathcal{ALC} KB $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent if there exists a model \mathcal{I} of \mathcal{A} and \mathcal{T} .

Given a fuzzy KB Σ , and a fuzzy assertion ψ (resp. a GCI $C \sqsubseteq D$), we say that Σ *entails* ψ (resp. $C \sqsubseteq D$), denoted $\Sigma \models \psi$ (resp. $\Sigma \models C \sqsubseteq D$), if each model of Σ is a model of ψ (resp. $C \sqsubseteq D$). Finally, given Σ and a fuzzy assertion α , it is of interest to compute α 's best lower and upper truth value bounds. The *greatest lower bound* of α w.r.t. Σ (denoted $glb(\Sigma, \alpha)$) is $glb(\Sigma, \alpha) = \sup\{n \mid \Sigma \models \langle \alpha \geq n \rangle\}$ where $\sup \emptyset = 0$. Similarly, the *least upper bound* of α w.r.t. Σ (denoted $lub(\Sigma, \alpha)$) is $lub(\Sigma, \alpha) = \inf\{n \mid \Sigma \models \langle \alpha \leq n \rangle\}$ where $\inf \emptyset = 1$. Determining the *glb* is called the *Best Truth Value Bound* (BTVB) problem. Basic inference problems are: (i) Check if a fuzzy KB is *consistent*, i.e. has a model. (ii) Check if D *subsumes* C w.r.t. Σ , i.e. $\Sigma \models C \sqsubseteq D$. (iii) Check if a is instance of C to degree $\geq n$, i.e. $\Sigma \models \langle a:C \geq n \rangle$ (Similarly for the other relations $\leq, >$ and $<$). (iv) Determine $glb(\Sigma, a:C)$.

We recall that all the inference problems can be reduced to the consistency problem [18]: (i) Concerning the entailment problem, it can be verified that it can be reduced to the inconsistency problem: $\langle \mathcal{T}, \mathcal{A} \rangle \models \langle \alpha \geq n \rangle$ iff $\langle \mathcal{T}, \mathcal{A} \cup \{\langle \alpha < n \rangle\} \rangle$ is inconsistent, and similarly for the other relations $\leq, >$ and $<$; (ii) Concerning the BTVB problem, it holds that $lub(\Sigma, a:C) = 1 - glb(\Sigma, a:\neg C)$, i.e. the *lub* can be determined through the *glb* (and vice-versa). Furthermore, the computation of the *glb* can be determined by relying on a finite number of entailment tests. First, for $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, we define $X^{\mathcal{A}} = \{0, 0.5, 1\} \cup \{n \mid \langle \alpha \bowtie n \rangle \in \mathcal{A}\}$ and $N^{\mathcal{A}} = X^{\mathcal{A}} \cup \{1 - n \mid n \in X^{\mathcal{A}}\}$. Then $glb(\Sigma, a:C) = \max\{n \mid n \in N^{\mathcal{A}} \text{ and } \Sigma \models \langle \alpha \geq n \rangle\}$. (iii) Concerning the subsumption problem, we have that $\Sigma \models C \sqsubseteq D$ iff $\Sigma = \langle \mathcal{T}, \{\langle a:C \geq n \rangle, \langle a:D < n \rangle\} \rangle$, with $n \in \{n_1, n_2\}$, $n_1 \in (0, 0.5]$ and $n_2 \in (0.5, 1]$ (e.g., we may choose $n_1 = 0.25, n_2 = 0.75$), is not consistent. So, the subsumption problem can be reduced to the consistency problem as well.

In all previous approaches to fuzzy DLs [18, 6, 14] decision procedures for the consistency, the entailment and the BTVB problem are given for various DL languages, but with restrictions on the form of concept inclusion axioms in a TBox \mathcal{T} . More precisely, \mathcal{T} was considered to be *simple*, i.e. cyclic axioms are not allowed, while concept inclusions were restricted to those of the form $A \sqsubseteq D$, where A is an *atomic* concept. Both GCIs and cyclic axioms are considered important for the classical case and, thus, should be provided in the fuzzy variant as well. For instance, cyclic definitions allow us to consider definitions such as

$$\begin{aligned} \text{Human} &\sqsubseteq \exists \text{hasParent}. \text{Human} \\ \text{GeometricElement} &\sqsubseteq \forall \text{hasPart}. \text{GeometricElement} . \end{aligned}$$

It is also well known that GCIs are used to express important features like: (i) The *domain* of a role R is concept C . This can be expressed by means of the GCI, $\exists R. \top \sqsubseteq C$. (ii) The *range* of a role R is concept C . This can be expressed by means of the GCI, $\top \sqsubseteq \forall R. C$ ⁵. (iii) Concept C_1 and concept C_2 are *disjoint*. This can be expressed by means of the GCI, $C_1 \sqcap C_2 \sqsubseteq \perp$.⁶ Such features appear in the OWL DL [9] and hence also in the fuzzy OWL DL language [16]. In the following we will present a calculus for fuzzy \mathcal{ALC} with both GCIs and cyclic axioms.

3 DEALING WITH GCIs IN FUZZY DLs

Suppose that we have an individual a and a concept C . Then, for any fuzzy interpretation \mathcal{I} , $\forall n \in [0, 1]_{\mathbb{Q}}$ either $C^{\mathcal{I}}(a^{\mathcal{I}}) < n$ or $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq n$ holds⁷. Furthermore, if $\mathcal{I} \models C \sqsubseteq D$, then either $C^{\mathcal{I}}(a^{\mathcal{I}}) < n$ or $D^{\mathcal{I}}(a^{\mathcal{I}}) \geq n$. From this observation, the following can be shown.

Proposition 1 $\mathcal{I} \models C \sqsubseteq D$ iff for all $n \in [0, 1]_{\mathbb{Q}}$, either $\mathcal{I} \models \langle a : C < n \rangle$ or $\mathcal{I} \models \langle a : D \geq n \rangle$, for all a .

This suggests that the models of a TBox \mathcal{T} can be captured in the form of mutually exclusive ABoxes. For example, if $\mathcal{T} = \{C_1 \sqsubseteq D_1, C_2 \sqsubseteq D_2\}$, then for any n , the four alternatives are (i) $\{\langle a:C_1 < n \rangle, \langle a:C_2 < n \rangle\}$; (ii) $\{\langle a:C_1 < n \rangle, \langle a:D_2 \geq n \rangle\}$; (iii) $\{\langle a:D_1 \geq n \rangle, \langle a:C_2 < n \rangle\}$; and (iv) $\{\langle a:D_1 \geq n \rangle, \langle a:D_2 \geq n \rangle\}$. Note that this is a generalisation of crisp DLs where for any crisp model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of \mathcal{T} , we have that $\forall d \in \Delta^{\mathcal{I}}. d \in ((\neg C_1 \sqcup D_1) \sqcap (\neg C_2 \sqcup D_2))^{\mathcal{I}}$. Please note that this *internalized* [2] concept cannot be used in fuzzy DLs since a GCI of the form $C \sqsubseteq D$ is not equivalent to the concept $\neg C \sqcup D$. Hence, in order to decide the consistency of a $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, for each individual a that exists in \mathcal{A} , or might be created by the reasoning algorithm, we have to create 2^k ABoxes, where k is the number of GCIs that exist in \mathcal{T} .

However, it is practically impossible to devise a terminating reasoning algorithm that uses Proposition 1 to handle GCIs and cyclic axioms as we cannot realistically apply it to all $n \in [0, 1]_{\mathbb{Q}}$. Fortunately, we can restrict these n to a *finite* set of values. Indeed, from the previous section, it turns out that the good candidate is the set $N^{\mathcal{A}}$. In [18, 19] it is shown that if a fuzzy \mathcal{ALC} ABox is consistent, then there exists a model where the membership degrees used to build the model are restricted to those that exist in the ABox. For instance,

⁵ Note that the top concept (\top) is not an atomic concept, hence range restrictions indeed are GCIs.

⁶ Note that $C \sqsubseteq \neg D$ is not a proper disjoint axiom in fuzzy \mathcal{ALC} .

⁷ Similarly, either $C^{\mathcal{I}}(a^{\mathcal{I}}) \leq n$ or $C^{\mathcal{I}}(a^{\mathcal{I}}) > n$.

in order to satisfy $\{\langle a:C \geq n \rangle\}$, we set $C^{\mathcal{I}}(a^{\mathcal{I}}) = n$, while to satisfy $\{\langle a:C > n \rangle\}$, we set $C^{\mathcal{I}}(a^{\mathcal{I}}) = n + \epsilon$, for a sufficiently small $\epsilon \in [0, 1]_{\mathbb{Q}}$.

In the following, we assume that an ABox \mathcal{A} has been *normalized*, i.e. fuzzy assertions of the form $\langle a:C > n \rangle$ are replaced by $\langle a:C \geq n + \epsilon \rangle$ and those of the form $\langle a:C < n \rangle$, by $\langle a:C \leq n - \epsilon \rangle$. Please note that in a normalized fuzzy KB we allow the degree to range in $[-\epsilon, 1 + \epsilon]_{\mathbb{Q}}$ in place of $[0, 1]_{\mathbb{Q}}$. It can be proved that the process of normalization is satisfiability preserving.

Proposition 2 Let $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ be a fuzzy knowledge base. Then Σ is satisfiable if and only if its normalized variant is satisfiable.

3.1 A fuzzy tableau for fuzzy \mathcal{ALC}

We have seen that the inference problems in fuzzy DLs can be reduced to the consistency checking problem. Similar to crisp DLs, our tableau algorithm checks the consistency of a fuzzy KB by trying to build a fuzzy tableau, from which it is immediate either to build a model in case KB is consistent or to detect that the KB is inconsistent. The fuzzy tableau we present here can be seen as an extension of the tableau presented in [8], and is inspired by the one presented in [14]. Without loss of generality, we assume that all concepts C are in *negation normal form* (NNF) [7], i.e. negations occur in front of atomic concepts only.⁸ In the following, $\bowtie \in \{\geq, \leq\}$, while we also use \bowtie^- to denote the *reflection* of \bowtie , e.g. if $\bowtie = \leq$, then $\bowtie^- = \geq$.

Definition 1 Given $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, let \mathbf{R}_{Σ} be the set of roles occurring in Σ and let $\text{sub}(\Sigma)$ be the set of named concepts appearing in Σ . A fuzzy tableau T for Σ is a quadruple $(\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V})$ such that: \mathbf{S} is a set of elements, $\mathcal{L} : \mathbf{S} \times \text{sub}(\Sigma) \rightarrow [0, 1]_{\mathbb{Q}}$ maps each element and concept, to a membership degree (the degree of the element being an instance of the concept), and $\mathcal{E} : \mathbf{R}_{\Sigma} \times 2^{\mathbf{S} \times \mathbf{S}} \rightarrow [0, 1]_{\mathbb{Q}}$ maps each role of \mathbf{R}_{Σ} and pair of elements to the membership degree of the pair being an instance of the role, and $\mathcal{V} : \mathbf{I}_{\mathcal{A}} \rightarrow \mathbf{S}$ maps individuals occurring in \mathcal{A} to elements in \mathbf{S} . For all $s, t \in \mathbf{S}$, $C, E \in \text{sub}(\Sigma)$, and $R \in \mathbf{R}_{\Sigma}$, T has to satisfy:

1. $\mathcal{L}(s, \perp) = 0$ and $\mathcal{L}(s, \top) = 1$ for all $s \in \mathbf{S}$,
2. If $\mathcal{L}(s, \neg C) \bowtie n$, then $\mathcal{L}(s, C) \bowtie^- 1 - n$.
3. If $\mathcal{L}(s, C \sqcap E) \geq n$, then $\mathcal{L}(s, C) \geq n$ and $\mathcal{L}(s, E) \geq n$.
4. If $\mathcal{L}(s, C \sqcup E) \leq n$, then $\mathcal{L}(s, C) \leq n$ and $\mathcal{L}(s, E) \leq n$.
5. If $\mathcal{L}(s, C \sqcup E) \geq n$, then $\mathcal{L}(s, C) \geq n$ or $\mathcal{L}(s, E) \geq n$.
6. If $\mathcal{L}(s, C \sqcap E) \leq n$, then $\mathcal{L}(s, C) \leq n$ or $\mathcal{L}(s, E) \leq n$.
7. If $\mathcal{L}(s, \forall R.C) \geq n$, then $\mathcal{E}(R, \langle s, t \rangle) \leq 1 - n$ or $\mathcal{L}(t, C) \geq n$ for all $t \in \mathbf{S}$.
8. If $\mathcal{L}(s, \exists R.C) \leq n$, then $\mathcal{E}(R, \langle s, t \rangle) \leq n$ or $\mathcal{L}(t, C) \leq n$ for all $t \in \mathbf{S}$.
9. If $\mathcal{L}(s, \exists R.C) \geq n$, then there exists $t \in \mathbf{S}$ such that $\mathcal{E}(R, \langle s, t \rangle) \geq n$ and $\mathcal{L}(t, C) \geq n$.
10. If $\mathcal{L}(s, \forall R.C) \leq n$, then there exists $t \in \mathbf{S}$ such that $\mathcal{E}(R, \langle s, t \rangle) \geq 1 - n$ and $\mathcal{L}(t, C) \leq n$.
11. If $C \sqsubseteq D \in \mathcal{A}$, then either $\mathcal{L}(s, C) \leq n - \epsilon$ or $\mathcal{L}(s, D) \geq n$, for all $s \in \mathbf{S}$ and $n \in N^{\mathcal{A}}$.
12. If $\langle a:C \bowtie n \rangle \in \mathcal{A}$, then $\mathcal{L}(\mathcal{V}(a), C) \bowtie n$.
13. If $\langle (a, b):R \bowtie n \rangle \in \mathcal{A}$, then $\mathcal{E}(R, \langle \mathcal{V}(a), \mathcal{V}(b) \rangle) \bowtie n$.

Proposition 3 $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent iff there exists a fuzzy tableau for Σ .

⁸ A fuzzy \mathcal{ALC} concept can be transformed into an equivalent one in NNF by pushing negations inwards using a combination of the De Morgan laws and the equivalences $\neg \exists R.C \equiv \forall R.\neg C$, $\neg \forall R.C \equiv \exists R.\neg C$.

Proof: [Sketch] For the *if* direction if $T = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V})$ is a fuzzy tableau for Σ , we can construct a fuzzy interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that is a model of \mathcal{A} and \mathcal{T} as follows:

$$\begin{aligned} \Delta^{\mathcal{I}} = \mathbf{S} \quad a^{\mathcal{I}} = \mathcal{V}(a), a \text{ occurs in } \mathcal{A} \quad A^{\mathcal{I}}(s) = \mathcal{L}(s, A) \text{ for all } s \in \mathbf{S} \\ \top^{\mathcal{I}}(s) = \mathcal{L}(s, \top), \perp^{\mathcal{I}}(s) = \mathcal{L}(s, \perp), \text{ for all } s \in \mathbf{S} \\ R^{\mathcal{I}}(s, t) = \mathcal{E}(R, \langle s, t \rangle) \text{ for all } \langle s, t \rangle \in \mathbf{S} \times \mathbf{S} \end{aligned}$$

To prove that \mathcal{I} is a model of \mathcal{A} and \mathcal{T} , we can show by induction on the structure of concepts that $\mathcal{L}(s, C) \bowtie n$ implies $C^{\mathcal{I}}(s) \bowtie n$ for all $s \in \mathbf{S}$. Together with properties 12, 13, Proposition 1, the fact that we can restrict our attention to the degrees in $N^{\mathcal{A}}$ and the interpretation of individuals and roles, this implies that \mathcal{I} is a model of \mathcal{T} , and that it satisfies each fuzzy assertion in \mathcal{A} .

For the converse, if $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of Σ , then a fuzzy tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V})$ for Σ can be defined as follows:

$$\mathbf{S} = \Delta^{\mathcal{I}} \quad \mathcal{E}(R, \langle s, t \rangle) = R^{\mathcal{I}}(s, t) \quad \mathcal{L}(s, C) = C^{\mathcal{I}}(s) \quad \mathcal{V}(a) = a^{\mathcal{I}}$$

It is easy to show that T is a fuzzy tableau for Σ . \square

3.2 An algorithm for constructing a fuzzy tableau

In order to decide the consistency of $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ a procedure that constructs a fuzzy tableau T for Σ has to be determined. Like the tableau algorithm presented in [8], our algorithm works on *completion-forests* since an ABox might contain several individuals with arbitrary roles connecting them. Due to the presence of general or cyclic terminologies, the termination of the algorithm is ensured by the use of *blocking*, where an expansion is terminated when individuals on the same path are asserted to belong to the same concepts.

Definition 2 Let $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ be a fuzzy KB. A completion-forest \mathcal{F} for Σ is a collection of trees whose distinguished roots are arbitrarily connected by edges. Each node x is labelled with a set $\mathcal{L}(x) = \{\langle C, \bowtie, n \rangle\}$, where $C \in \text{sub}(\Sigma)$, $\bowtie \in \{\geq, \leq\}$ and $n \in [-\epsilon, 1 + \epsilon]$. Each edge $\langle x, y \rangle$ is labelled with a set $\mathcal{L}(\langle x, y \rangle) = \{\langle R, \bowtie, n \rangle\}$, where $R \in \mathbf{R}_{\Sigma}$ are roles occurring in Σ . Two triples $\langle C, \geq, n \rangle$ ($\langle R, \geq, n \rangle$) and $\langle C, \leq, m \rangle$ ($\langle R, \leq, m \rangle$) are conjugated if $n > m$.

If nodes x and y are connected by an edge $\langle x, y \rangle$ with $\langle R, \bowtie, n \rangle \in \mathcal{L}(\langle x, y \rangle)$, then y is called an $R_{\bowtie n}$ -successor of x and x is called an $R_{\bowtie n}$ -predecessor of y . Let y be an $R_{\geq n}$ -successor of x , the edge $\langle x, y \rangle$ is conjugated with triples $\langle R, \leq, m \rangle$ if $n > m$. Similarly, we can extend it to the cases of $R_{\leq n}$ -successor. A node x is an R -successor (resp. R -predecessor) of y if it is an $R_{\bowtie n}$ -successor (resp. $R_{\bowtie n}$ -predecessor) of y for some role R . As usual, ancestor is the transitive closure of predecessor.

A node x is directly blocked iff none of its ancestors are blocked, it is not a root node, and it has an ancestor y such that $\mathcal{L}(x) \subseteq \mathcal{L}(y)$. In this case, we say y directly blocks x . A node x is blocked iff it is directly blocked or if one of its predecessor is blocked.

A node x is said to contain a clash iff there exist two conjugated triples in $\mathcal{L}(x)$ or one of the following triples exists within $\mathcal{L}(x)$: (i) $\langle \perp, \geq, n \rangle$, for $n > 0$; (ii) $\langle \top, \leq, n \rangle$, for $n < 1$; (iii) $\langle C, \leq, -\epsilon \rangle$; (iv) $\langle C, \geq, 1 + \epsilon \rangle$. The notion of ' $\mathcal{L}(\langle x, y \rangle)$ contains a clash' is defined similarly.

The algorithm initializes a forest \mathcal{F} to contain (i) a root node x_0^i , for each individual a_i occurring in \mathcal{A} , labelled with $\mathcal{L}(x_0^i)$ such that $\{\langle C_i, \bowtie, n \rangle\} \subseteq \mathcal{L}(x_0^i)$ for each fuzzy assertion $\langle a_i:C_i \bowtie n \rangle \in \mathcal{A}$, and (ii) an edge $\langle x_0^i, x_0^j \rangle$, for each fuzzy assertion $\langle (a_i, a_j):R_i \bowtie n \rangle \in \mathcal{A}$, labelled with $\mathcal{L}(\langle x_0^i, x_0^j \rangle)$ such that $\{\langle R_i, \bowtie, n \rangle\} \subseteq \mathcal{L}(\langle x_0^i, x_0^j \rangle)$. \mathcal{F} is then expanded by repeatedly

Table 2. Tableaux expansion rules

Rule	Description	Rule	Description
(\neg)	if 1. $\langle \neg C, \bowtie, n \rangle \in \mathcal{L}(x)$, and 2. $\langle C, \bowtie^-, 1-n \rangle \notin \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C, \bowtie^-, 1-n \rangle\}$	(\exists_{\geq})	if 1. $\langle \exists R.C, \geq, n \rangle \in \mathcal{L}(x)$, x is not blocked, 2. x has no $R_{\geq n}$ -successor y with $\langle C, \geq, n \rangle \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{\langle R, \geq, n \rangle\}$, $\mathcal{L}(y) = \{\langle C, \geq, n \rangle\}$,
(\sqcap_{\geq})	if 1. $\langle C_1 \sqcap C_2, \geq, n \rangle \in \mathcal{L}(x)$ and 2. $\{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\}$	(\forall_{\leq})	if 1. $\langle \forall R.C, \leq, n \rangle \in \mathcal{L}(x)$, x is not blocked, 2. x has no $R_{\geq 1-n}$ -successor y with $\langle C, \leq, n \rangle \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{\langle R, \geq, 1-n \rangle\}$, $\mathcal{L}(y) = \{\langle C, \leq, n \rangle\}$,
(\sqcup_{\leq})	if 1. $\langle C_1 \sqcup C_2, \leq, n \rangle \in \mathcal{L}(x)$ and 2. $\{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\}$	(\forall_{\geq})	if 1. $\langle \forall R.C, \geq, n \rangle \in \mathcal{L}(x)$, 2. x has an R -successor y with $\langle C, \geq, n \rangle \notin \mathcal{L}(y)$ and 3. $\langle R, \leq, 1-n \rangle$ is conjugated with the edge $\langle x, y \rangle$ then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle C, \geq, n \rangle\}$,
(\sqcup_{\geq})	if 1. $\langle C_1 \sqcup C_2, \geq, n \rangle \in \mathcal{L}(x)$ and 2. $\{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\}$	(\exists_{\leq})	if 1. $\langle \exists R.C, \leq, n \rangle \in \mathcal{L}(x)$, 2. x has an R -successor y with $\langle C, \leq, n \rangle \notin \mathcal{L}(y)$ and 3. $\langle R, \leq, n \rangle$ is conjugated with the edge $\langle x, y \rangle$ then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle C, \leq, n \rangle\}$,
(\sqcap_{\leq})	if 1. $\langle C_1 \sqcap C_2, \leq, n \rangle \in \mathcal{L}(x)$ and 2. $\{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\}$	(\sqsubseteq)	if 1. $C \sqsubseteq D \in \mathcal{T}$ and 2. $\{\langle C, \leq, n-\epsilon \rangle, \langle D, \geq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ for $n \in N^A$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$ for some $E \in \{\langle C, \leq, n-\epsilon \rangle, \langle D, \geq, n \rangle\}$

applying the completion rules from Table 2. The completion forest is complete when, for some node x (edge $\langle x, y \rangle$), $\mathcal{L}(x)$ ($\mathcal{L}(\langle x, y \rangle)$) contains a clash, or none of the completion rules in Table 2 are applicable. The algorithm stops when a clash occurs; it answers ‘ Σ is consistent’ iff the completion rules can be applied in such a way that they yield a complete and clash-free completion forest, and ‘ Σ is inconsistent’ otherwise.

From Table 2, we can see that for an arbitrary fuzzy assertion of the form $\langle a:D \bowtie n \rangle$ either value n or its complement $1-n$ appear in the expansion of a node x where $D \in \mathcal{L}(x)$. The finite property of the membership degrees makes blocking possible in our algorithm.

Example 1 Let us show how the blocking condition works on the cyclic fuzzy KB, $\Sigma = \{\langle \text{HotPinkRose} \sqsubseteq \exists \text{nextGen.HotPinkRose} \rangle, \{\langle a:\text{HotPinkRose} \geq 0.6 \rangle\}\}$. Σ is satisfiable and $N^A = \{0, 0.5, 1\} \cup \{0.4, 0.6\}$. We start with a root node x^a , with label $\mathcal{L}(x^a) = \{\langle \text{HotPinkRose}, \geq, 0.6 \rangle\}$. By applying rule (\sqsubseteq) to node x^a , to $\text{HotPinkRose} \sqsubseteq \exists \text{nextGen.HotPinkRose}$ with, e.g., $n = 0.6$, and $E = \langle \exists \text{nextGen.HotPinkRose}, \geq, n \rangle$, we update the label $\mathcal{L}(x^a)$ with $\mathcal{L}(x^a) = \{\langle \text{HotPinkRose}, \geq, 0.6 \rangle, \langle \exists \text{nextGen.HotPinkRose}, \geq, 0.6 \rangle\}$. Continuing with node x^a , we apply rule (\exists_{\geq}) to $\langle \exists \text{nextGen.HotPinkRose}, \geq, 0.6 \rangle$, create a new edge $\langle x^a, y_1 \rangle$ with $\mathcal{L}(\langle x^a, y_1 \rangle) = \{\langle \text{nextGen}, \geq, 0.6 \rangle\}$ and $\mathcal{L}(y_1) = \{\langle \text{HotPinkRose}, \geq, 0.6 \rangle\}$. By continuing with node y_1 exactly as for node x^a , after applying rule (\sqsubseteq) , we update the label $\mathcal{L}(y_1)$ with $\mathcal{L}(y_1) = \{\langle \text{HotPinkRose}, \geq, 0.6 \rangle, \langle \exists \text{nextGen.HotPinkRose}, \geq, 0.6 \rangle\}$. Now, y_1 is an nextGen -successor of x^a and $\mathcal{L}(y_1) = \mathcal{L}(x^a)$ and, thus, y_1 is directly blocked.

Example 2 We show that $\Sigma = \{\{C \sqsubseteq D\}, \{\langle a:C > 0.3 \rangle, \langle a:D \leq 0.3 \rangle\}\}$ is inconsistent. We first normalize Σ into $\Sigma = \{\{C \sqsubseteq D\}, \{\langle a:C \geq 0.3 + \epsilon \rangle, \langle a:D \leq 0.3 \rangle\}\}$, for a small $\epsilon > 0$, e.g. $\epsilon = 0.01$. $N^A = \{0, 0.5, 1\} \cup \{0.3, 0.3 + \epsilon, 0.7 - \epsilon, 0.7\}$. We start with a root node x^a and $\mathcal{L}(x^a) = \{\langle C, \geq, 0.3 + \epsilon \rangle, \langle D, \leq, 0.3 \rangle\}$. By applying rule (\sqsubseteq) to node x^a , to $C \sqsubseteq D$ with, e.g., $n = 0.3 + \epsilon$ we get two branches, depending on the ‘choice of $E \in \{\langle C, \leq, 0.3 + \epsilon \rangle, \langle D, \geq, 0.3 + \epsilon \rangle\}$ ’. In the former case, we update the label $\mathcal{L}(x^a)$ with $\mathcal{L}(x^a) = \{\langle C, \geq, 0.3 + \epsilon \rangle, \langle D, \leq, 0.3 \rangle, \langle C, \leq, 0.3 \rangle\}$

which contains a clash, while in the latter case we update the label $\mathcal{L}(x^a)$ with $\mathcal{L}(x^a) = \{\langle C, \geq, 0.3 + \epsilon \rangle, \langle D, \leq, 0.3 \rangle, \langle D, \geq, 0.3 + \epsilon \rangle\}$ which also contains a clash. No complete, clash-free forest can be obtained, thus the algorithm answers with ‘inconsistent’.

Proposition 4 (Termination) For each fuzzy ALC KB Σ , the tableau algorithm terminates.

Proof: [Sketch] Termination is a result of the properties of the expansion rules, as in the classical case [8]. More precisely we have the following observations. (i) The expansion rules never remove nodes from the tree or concepts from node labels or change the edge labels. (ii) Successors are only generated by the rules \exists_{\geq} and \forall_{\leq} . For any node and for each concept these rules are applied at-most once. (iii) Since nodes are labelled with nonempty subsets of $\text{sub}(\Sigma)$, obviously there is a finite number of possible labellings for a pair of nodes. Thus, the condition of blocking will be applied in any path of the tree and consequently any path will have a finite length. \square

Proposition 5 (Soundness) If the expansion rules can be applied to an fuzzy ALC KB Σ such that they yield a complete and clash-free completion-forest, then Σ has a fuzzy tableau for Σ .

Proof: [Sketch] Let \mathcal{F} be a complete and clash-free completion-forest constructed by the tableaux algorithm for Σ . A fuzzy tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V})$ can be defined as follows:

$$\begin{aligned}
 \mathbf{S} &= \{x \mid x \text{ is a node in } \mathcal{F}, \text{ and } x \text{ is not blocked}\}, \\
 \mathcal{L}(x, \perp) &= 0, \text{ for all } x \in \mathbf{S}, \\
 \mathcal{L}(x, \top) &= 1, \text{ for all } x \in \mathbf{S}, \\
 \mathcal{L}(x, C) &= \max[\langle C, \geq, n_i \rangle], \text{ for all } x \in \mathcal{F} \text{ not blocked}, \\
 \mathcal{L}(x, \neg A) &= 1 - \mathcal{L}(x, A), \text{ for all } x \in \mathcal{F} \text{ not blocked}, \\
 &\quad \text{with } \langle \neg A, \geq, n \rangle \in \mathcal{L}(x), \\
 \mathcal{E}(R, \langle x, y \rangle) &= \{\max[\langle R, \geq, n_i \rangle] \mid \\
 &\quad 1. y \text{ is an } R_{\geq n_i} \text{-successor of } x \text{ or} \\
 &\quad 2. \langle R, \geq, n_i \rangle \in \mathcal{L}(\langle x, z \rangle) \text{ and } y \text{ blocks } z\}, \\
 \mathcal{V}(a_i) &= x_0^i, \text{ where } x_0^i \text{ is a root node}
 \end{aligned}$$

where \max returns the maximum degree n out of the set of triples of the form $\langle A, \geq, n_i \rangle$, or 0 if no such triple exists. It can be shown that T is a fuzzy tableau for Σ . \square

Proposition 6 (Completeness) Consider a fuzzy ALC KB Σ . If Σ has a fuzzy tableau, then the expansion rules can be applied in such a way that the tableaux algorithm yields a complete and clash-free completion-forest for Σ .

Proof: [Sketch] Let $T = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V})$ be a fuzzy tableau for Σ . Using T , we trigger the application of the expansion rules such that they yield a completion-forest \mathcal{F} that is both complete and clash-free. Similarly to [8] we can define a mapping π which maps nodes of \mathcal{F} to elements of \mathbf{S} , and guide the application of the non-deterministic rules \sqsubseteq , \sqcup_{\geq} and \sqcap_{\leq} . Our method slightly differs from the one used in crisp DLs [8] in the following way. Using the membership degree of a node to a concept, found in the fuzzy tableau, we create artificial triples that are tested against conjugation with the candidate triples that the non-deterministic rule can insert in the completion-forest. The triples that don't cause a conjugation can be added. The modified rules, which are used to guide such an expansion, are presented in Table 3. The modified rules together with the termination property ensure the completeness of the algorithm. \square

Table 3. The \sqsubseteq' , \sqcup'_{\geq} and \sqcap'_{\leq} rules

(\sqsubseteq')	if 1. $C \sqsubseteq D \in \mathcal{T}$, x is not indirectly blocked and 2. $\{\langle C, \leq, n - \epsilon \rangle, \langle D, \geq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ for $n \in N^A$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$ for some $E \in \{\langle C, \leq, n - \epsilon \rangle, \langle D, \geq, n \rangle\}$, not conjugated with $\langle C, \geq, \mathcal{L}(\pi(x), C) \rangle$ or $\langle D, \leq, \mathcal{L}(\pi(x), D) \rangle$
(\sqcup'_{\geq})	if 1. $\langle C_1 \sqcup C_2, \geq, n \rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \geq, n \rangle, \langle C_2, \geq, n \rangle\}$ not conjugated with $\langle C_1, \leq, \mathcal{L}(\pi(x), C_1) \rangle$ or $\langle C_2, \leq, \mathcal{L}(\pi(x), C_2) \rangle$
(\sqcap'_{\leq})	if 1. $\langle C_1 \sqcap C_2, \leq, n \rangle \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{\langle C_1, \leq, n \rangle, \langle C_2, \leq, n \rangle\}$ not conjugated with $\langle C_1, \geq, \mathcal{L}(\pi(x), C_1) \rangle$ or $\langle C_2, \geq, \mathcal{L}(\pi(x), C_2) \rangle$

4 CONCLUSIONS

Fuzzy DLs extend crisp DLs to deal with vague concepts. None of the work on fuzzy DLs so far presented a correct and complete calculus for cyclic TBoxes and general concept inclusions, which are important features of current crisp DL systems. We overcome to this limitation by providing a tableau for fuzzy ALC with GCIs.

Major topics for future research are indeed the extension of the fuzzy tableau algorithm to expressive DL languages such as fuzzy SHIF(D) or SHOIN(D) [21] and the development of a system supporting this language. In the former case, such algorithm can be based directly on the ones presented for the fuzzy ST and SHIN DLs [14, 15] and the rules for nominals, for SHOIN [10] and for fuzzy SHOIN [16].

ACKNOWLEDGEMENTS

The work of Giorgos Stoilos, Giorgos Stamou and Jeff Z. Pan was partially supported by the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507482).

REFERENCES

- [1] M. d'Aquin, J. Lieber, and A. Napoli, 'Towards a semantic portal for oncology using a description logic with fuzzy concrete domains', in *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, ed., Elie Sanchez, Elsevier, 2006. To appear.
- [2] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, Cambridge University Press, 2003.
- [3] P. Bonatti and A. Tettamanzi, 'Some complexity results on fuzzy description logics', in *WILF 2003 International Workshop on Fuzzy Logic and Applications*, LNCS 2955, Berlin, 2004. Springer Verlag.
- [4] P. Hájek, 'Making fuzzy description logics more expressive', *Fuzzy Sets and Systems*, 2005.
- [5] S. Hölldobler, N. H. Nga, and T. D. Khang, 'The fuzzy description logic ALC_{FH}', in *Proc. of the Int. Workshop on Description Logics*, 2005.
- [6] S. Hölldobler, H. -P. Störr, and T. D. Khang, 'The fuzzy description logic ALC_{FH} with hedge algebras as concept modifiers', *Journal of Advanced Computational Intelligence*, 2003.
- [7] B. Hollunder, W. Nutt, and M. Schmidt-Schaub, 'Subsumption algorithms for concept description languages', in *Proc. of ECAI-90, 9th European Conf. on Artificial Intelligence*, pp. 348–353, 1990.
- [8] I. Horrocks, U. Sattler, and S. Tobies, 'Reasoning with individuals for the description logic SHIQ', in *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, ed., David MacAllester, LNAI 1831, pp. 482–496, Germany, 2000. Springer Verlag.
- [9] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, 'From SHIQ and RDF to OWL: The making of a web ontology language', *Journal of Web Semantics*, 1(1), 7–26, 2003.
- [10] I. Horrocks and U. Sattler, 'A Tableaux Decision Procedure for SHOTQ', *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.
- [11] Y. Li, B. Xu, J. Lu, D. Kang, and P. Wang, 'Extended fuzzy description logic ALCN', in *Proc. of the 9th Int. Conference, Knowledge-Based Intelligent Information and Engineering Systems (KES-05)*, LNCS 3684, pp. 896–902. Springer Verlag, 2005.
- [12] D. Sanchez and A. G.B. Tettamanzi, 'Fuzzy quantification in fuzzy description logics', in *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, ed., Elie Sanchez, Elsevier, 2006. To appear.
- [13] M. Schmidt-Schaub and Gert Smolka, 'Attributive concept descriptions with complements', *Artificial Intelligence*, 48, 1–26, 1991.
- [14] G. Stoilos, G. Stamou, V. Tzouvaras, Jeff Z. Pan, and I. Horrocks, 'A Fuzzy Description Logic for Multimedia Knowledge Representation', *Proc. of the Int. Workshop on Multimedia and the Semantic Web*, 2005.
- [15] G. Stoilos, G. Stamou, V. Tzouvaras, J. Pan, and I. Horrocks, 'The fuzzy description logic f-SHIN'. Int. Workshop on Uncertainty Reasoning For the Semantic Web, in ISWC '05, Galway, 2005.
- [16] G. Stoilos, G. Stamou, V. Tzouvaras, J. Pan, and I. Horrocks, 'Fuzzy OWL: Uncertainty and the semantic web'. Int. Workshop of OWL: Experiences and Directions, Galway, 2005.
- [17] U. Straccia, 'A framework for the retrieval of multimedia objects based on four-valued fuzzy description logics', in *Soft Computing in Information Retrieval: Techniques and Applications*, 332–357, Physica Verlag (Springer Verlag), Heidelberg, Germany, 2000.
- [18] U. Straccia, 'Reasoning within fuzzy description logics', *Journal of Artificial Intelligence Research*, 14, 137–166, 2001.
- [19] U. Straccia, 'Transforming fuzzy description logics into classical description logics', in *Proc. of the 9th European Conf. on Logics in Artificial Intelligence (JELIA-04)*, LNCS 3229, pp. 385–399, Lisbon, Portugal, 2004. Springer Verlag.
- [20] U. Straccia, 'Description logics with fuzzy concrete domains', in *21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pp. 559–567, Edinburgh, Scotland, 2005. AUAI Press.
- [21] U. Straccia, 'Towards a fuzzy description logic for the semantic web (preliminary report)', in *2nd European Semantic Web Conference (ESWC-05)*, LNCS 3532, pp. 167–181, Crete, 2005. Springer Verlag.
- [22] U. Straccia, 'Uncertainty and description logic programs over lattices', in *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, ed., Elie Sanchez, Elsevier, 2006. To appear.
- [23] C. Tresp and R. Molitor, 'A description logic for vague knowledge', in *Proc. of the 13th European Conf. on Artificial Intelligence*, 1998.
- [24] J. Yen, 'Generalizing term subsumption languages to fuzzy logic', in *Proc. of 12th Int. Joint Conf. on Artificial Intelligence*, pp. 472–477, 1991.
- [25] L. A. Zadeh, 'Fuzzy sets', *Information and Control*, 8(3), 338–353, 1965.
- [26] L. Zhang, Y. Yu, J. Zhou, C. Lin, and Y. Yang, 'An enhanced model for searching in semantic portals', in *WWW '05: Proc. of the 14th Int. Conference on World Wide Web*, pp. 453–462, USA, 2005. ACM Press.

Approximating Extended Answer Sets

Davy Van Nieuwenborgh¹ and Stijn Heymans² and Dirk Vermeir³

Abstract. We present an approximation theory for the extended answer set semantics, using the concept of an approximation constraint. Intuitively, an approximation constraint, while satisfied by a “perfect” solution, may be left unsatisfied in an approximate extended answer set. Approximations improve as the number of unsatisfied constraints decreases. We show how the framework can also capture the classical answer set semantics, thus providing an approximative version of the latter.

1 Introduction

The idea behind the answer set semantics for logic programs is both intuitive and elegant. Given a program P and a candidate answer set M , one computes a reduct program P_M of a simpler type for which a semantics P_M^* is known. The reduct P_M is obtained from P by taking into account the consequences of accepting the proposed truth values of the literals in M . The candidate set M is then an answer set just when $P_M^* = M$, i.e. M is “self-producible”.

In [17], the same reduction technique is applied to define the extended answer set semantics for simple logic programs, i.e. programs containing only classical negation. In contrast with the classical answer set semantics, extended answer sets are able to handle contradictory programs by allowing rules to be defeated, i.e. to leave certain rules unsatisfied.

The answer set semantics has been implemented in state-of-the-art solvers such as DLV[11] or SMODELS[14]. While these solvers perform well in many cases, they implement an “all or nothing” strategy, i.e. they do not provide the user with “partial” or “approximate” answer sets if there are insufficient resources for computing an exact solution. Clearly, given the inherent complexity of the problems tackled by such solvers, it would be desirable to have such an approximation facility, e.g. in cases where the available resources are limited. Furthermore, for many application areas obtaining such a “good enough” solution within a certain time limit, may be attractive, e.g. in diagnostic reasoning [18].

In this paper, we propose to provide an approximation theory for the extended answer set semantics, based on so-called approximation constraints. Intuitively, an approximation constraint is a constraint that should be satisfied by a perfect solution to a problem, but which may be left unsatisfied by an approximate solution. The more approximation constraints an approximate extended answer set satisfies, the “better” it approximates an exact solution. We will show that the proposed framework can also be used to approximate classical answer

sets⁴, which may lead to the development of “anytime” approximate answer set solvers.

The computation of extended answer sets for programs without constraints is very efficient, i.e. it takes quadratic time in the size of the program to compute an extended answer set for such programs, which implies that a first approximation can be computed very quickly. Although grounding adds an exponential⁵ factor [6] to the computation in the worst case, the extended answer set semantics should perform much better in the average case. Moreover, the semantics allows for a “ground when needed” implementation, in contrast to solvers such as DLV [11] or SMODELS [14] which first completely ground the program, before computing a solution.

The above mentioned efficiency implies that the basic extended answer set semantics is not able to handle problems located above \mathcal{P} . However, adding approximation constraints to the semantics lifts the complexity up to include \mathcal{NP} -complete problems, placing the proposed semantics at the same level as the classical answer set semantics.

The remainder of the paper is organized as follows: Section 2 introduces the extended answer set semantics, while Section 3 presents the framework of approximation constraints. In Section 4 the relationship with classical answer programming is explored, while we briefly discuss some related work in Section 5. Finally, we conclude and give some directions for future research in Section 6. Due to space restrictions, all proofs are omitted, they can be found in the technical report [16].

2 Extended Answer Set Semantics

In this section, we introduce the extended answer set semantics for simple logic programs, i.e. logic programs with only classical negation, no disjunction in the head of rules and no constraints.

A *term* is a constant or a variable, where the former will be written lower-case and the latter upper-case. An *atom* is of the form $p(t_1, \dots, t_n)$, $0 \leq n < \infty$, where p is an n -ary⁶ predicate name and t_i , $1 \leq i \leq n$, are terms. A *literal* is an atom a or a classically negated atom $\neg a$.

Definition 1 A *simple logic program (SLP)* is a countable set P of rules of the form $a \leftarrow \alpha$ where $\{a\} \cup \alpha$ is a finite set of literals⁷.

⁴ Conversely, Theorem 4 in [17] shows that classical answer set programming can be used to compute extended answer set semantics.

⁵ If we restrict the arities of predicates to small numbers, e.g. binary or ternary, it is shown in [8] that the additional complexity of grounding only adds one level of the polynomial hierarchy in contrast to the exponential factor in general.

⁶ We thus allow for 0-ary predicates, i.e., *propositions*.

⁷ Note that constraints, i.e. rules with empty heads, are not allowed for the moment.

¹ Supported by the Flemish Fund for Scientific Research (FWO-Vlaanderen).

² Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria. Email: stijn.heymans@deri.org

³ Department of Computer Science, Vrije Universiteit Brussel (VUB), Pleinlaan 2, B1050 Brussels, Belgium. Email: {dvnieuwe,dvermeir}@vub.ac.be

A *ground* atom, literal, rule, or SLP does not contain variables. Substituting every variable in a SLP P with every possible constant in P yields the ground SLP $gr(P)$.

Example 1 Grounding the SLP

$$\neg p(c, a) \leftarrow \quad p(X) \leftarrow \neg q(X, b) \quad q(X) \leftarrow \neg p(X, a)$$

yields the SLP⁸.

$$\begin{array}{lll} p(a) \leftarrow \neg q(a, b) & p(b) \leftarrow \neg q(b, b) & p(c) \leftarrow \neg q(c, b) \\ q(a) \leftarrow \neg p(a, a) & q(b) \leftarrow \neg p(b, a) & q(c) \leftarrow \neg p(c, a) \\ \neg p(c, a) \leftarrow & & \end{array}$$

In the rest of the paper we always assume ground SLPs and ground literals; to obtain the definitions for ungrounded SLPs, replace every occurrence of a SLP P by $gr(P)$, e.g., an extended answer set of an ungrounded SLP P is an extended answer set of $gr(P)$.

For a set of literals X , we use $\neg X$ to denote the set $\{\neg p \mid p \in X\}$ where $\neg\neg a \equiv a$. Further, X is said to be *consistent* if $X \cap \neg X = \emptyset$, i.e. X does not contain contradictory literals a and $\neg a$.

Definition 2 The *Herbrand base* \mathcal{B}_P of a SLP P is the set of all ground atoms that can be formed using the language of P . The set of all literals that can be formed with P , i.e. $\mathcal{B}_P \cup \neg \mathcal{B}_P$, is denoted by \mathcal{L}_P . An *interpretation* I of P is any consistent subset of \mathcal{L}_P .

A rule $r = a \leftarrow \alpha$ is **satisfied** by an interpretation I , denoted $I \models r$, if $a \in I$ whenever $\alpha \subseteq I$, i.e., if r is **applicable** ($\alpha \subseteq I$) then it must be **applied** ($\alpha \cup \{a\} \subseteq I$). The rule r is **defeated** w.r.t. I iff there exists an applied competing rule $\neg a \leftarrow \alpha'$ in P , such a rule is said to **defeat** r .

Intuitively, in an extended answer set (see below), a rule $r: a \leftarrow \alpha$ may not be left unsatisfied, unless one accepts the opposite conclusion $\neg a$ which must itself be motivated by a competing applied rule $\neg a \leftarrow \alpha'$ that defeats r .

Example 2 Consider the SLP P_1 containing the rules $\neg a \leftarrow$, $\neg b \leftarrow$, $a \leftarrow \neg b$, and $b \leftarrow \neg a$. For the interpretation $I = \{\neg a, b\}$ we have that I satisfies all rules in P_1 but one: $\neg a \leftarrow$ and $b \leftarrow \neg a$ are applied while $a \leftarrow \neg b$ is not applicable. The unsatisfied rule $\neg b \leftarrow$ is defeated by $b \leftarrow \neg a$.

For a set of rules P , we use P^* to denote the unique minimal [15] model of the positive logic program consisting of the rules in P , where negative literals $\neg a$ are considered as fresh atoms. We can compute P^* using the immediate consequence operator $T_P(X) = \{l \in \mathcal{L}_P \mid l \leftarrow \beta \in P \wedge \beta \subseteq X\}$. Clearly, the operator T_P is monotonic, and $T_P^\infty(\emptyset) = P^*$.

For the program of Example 2, we have that $P_1^* = \{a, b, \neg a, \neg b\}$ is inconsistent. The following definition allows us to not apply certain rules when computing a consistent interpretation for programs such as P_1 .

Definition 3 The *reduct* $P_I \subseteq P$ of a SLP P w.r.t. a set of literals I contains just the rules satisfied by I , i.e. $P_I = \{r \in P \mid I \models r\}$.

An interpretation S is an **extended answer set** of P iff $P_S^* = S$, i.e., S is **founded**, and all rules in $P \setminus P_S$ are defeated w.r.t. S .

We use $\mathcal{ES}(P)$ to denote the set of all extended answer sets of P .

⁸ Note that a variable X in a rule should be grounded with the same constant in that rule (either with a , b or c), while it may be grounded with other constants in other rules, i.e., the variables in a rule are considered local to the rule. One can, e.g., replace the above SLP by the equivalent program

$$\neg p(c, a) \leftarrow \quad p(X) \leftarrow \neg q(X, b) \quad q(Y) \leftarrow \neg p(Y, a)$$

Thus, the extended answer set semantics deals with inconsistencies in a simple yet intuitive way: when faced with contradictory applicable rules, just select one for application and ignore (defeat) the other. In the absence of extra information (e.g., a preference relation for satisfying certain rules at the expense of others [17]; or the approximation constraints introduced in this paper), this seems a reasonable strategy for extracting a consistent semantics out of inconsistent programs.

Reconsidering Example 2, it is easy to verify that P_1 has three extended answer sets, i.e. $M_1 = \{\neg a, b\}$, $M_2 = \{a, \neg b\}$ and $M_3 = \{\neg a, \neg b\}$. Note that e.g. $P_{M_1} = P_1 \setminus \{\neg b \leftarrow\}$, i.e. $\neg b \leftarrow$ is defeated w.r.t. M_1 .

While Definition 3 allows P_M , with M an extended answer set, to be a strict subset of P , it still maximizes the set of satisfied rules w.r.t. an extended answer set.

Theorem 1 Let P be a SLP and let M be an extended answer set for P . Then P_M is maximal w.r.t. set inclusion among the reducts of founded interpretations of P .

The reverse of the above theorem does not hold in general, as witnessed by the following example.

Example 3 Consider the program P_2 containing the following rules.

$$\neg a \leftarrow \quad b \leftarrow \quad \neg b \leftarrow \neg a$$

The interpretation $N = \{b\}$ is founded with $P_{2N} = \{b \leftarrow, \neg b \leftarrow \neg a\}$ which is clearly maximal since P_2^* is inconsistent. Still, N is not an extended answer set because $\neg a \leftarrow$ is not defeated.

The extended answer set semantics is universal.

Theorem 2 Every SLP has extended answer sets.

Moreover, the extended answer sets can be efficiently computed using an immediate consequence operator:

$$T_P^e(S) = \begin{cases} S \cup \{l\} & \text{if } r : l \leftarrow \alpha \in P \text{ and } \alpha \subseteq S \text{ and } \neg l \notin S, \\ S & \text{otherwise.} \end{cases}$$

Clearly, the above operator is non-deterministic as the result depends on the order in which rules are applied. However, it can be shown that $T_P^e(\emptyset)$ always results in an extended answer set of P . The non-deterministic behavior of T_P^e greatly influences the complexity of its implementation. In the worst case an algorithm for T_P^e will run in time quadratic⁹ in the size of the program, but on average, depending on the order in which rules get applied, the running time will be much better¹⁰.

Finally, note that the above implies that an implementation that uses T_P^e does not need a fully grounded version of the input program but may instead operate on a “ground (a rule) as needed” basis, as illustrated by the following example.

⁹ Note that, by a well-known result [6], grounding of the program may add an exponential (worst-case) factor to the total cost of computing an extended answer set for an ungrounded program. However, if the arities of the predicates are bounded by a small number, [8] shows that this exponential factor reduces to one level of the polynomial hierarchy.

¹⁰ Also in the non-grounded case the average complexity will be much better than the exponential worst-case scenario as, depending on the sequence in which rules get applied, certain parts of the program need not to be (fully) grounded.

Example 4 Consider the following non-grounded program.

$$\begin{array}{ll} p(a) \leftarrow & q(a) \leftarrow \\ p(b) \leftarrow & q(b) \leftarrow \\ \neg q(X) \leftarrow p(X) & \neg p(X) \leftarrow q(X) \end{array}$$

A possible computation of $T_P^{e\infty}(\emptyset)$ could result in the extended answer set $\{p(a), p(b), \neg q(a), \neg q(b)\}$, e.g. by starting the computation with applying the fact rule $p(a) \leftarrow$, i.e. $T_P^e(\emptyset) = \{p(a)\}$. As a consequence, we can make a ground instantiation $\neg q(a) \leftarrow p(a)$ of the rule $\neg q(X) \leftarrow p(X)$, which an implementation can choose to apply in computing $T_P^e(\{p(a)\}) = \{p(a), \neg q(a)\}$. If we now do a similar reasoning for the fact $p(b) \leftarrow$ to obtain $T_P^e(\{p(a), \neg q(a)\}) = \{p(a), \neg q(a), p(b)\}$ and $T_P^e(\{p(a), \neg q(a), p(b)\}) = T_P^{e\infty}(\emptyset) = \{p(a), p(b), \neg q(a), \neg q(b)\}$, it is easy to see that an implementation does not need any grounded version of the rule $\neg p(X) \leftarrow q(X)$, while classical implementations that first ground everything will obtain two grounded versions $\neg p(a) \leftarrow q(a)$ and $\neg p(b) \leftarrow q(b)$.

Note that we do not conjecture that our approach is better in the worst-case scenario. We only argue that by interleaving the grounding and computation process, one can get better results, both in time and space, in the average case.

3 Approximation Constraints

Because of Theorem 2, checking whether a SLP has an extended answer set can be done in constant time, and thus the semantics is not very expressive when compared to e.g. classical answer set semantics (for which the problem is \mathcal{NP} -complete). In this section, we increase the expressiveness by allowing for constraints to appear in programs.

Intuitively, an *approximation constraint* is like a traditional constraint, i.e. a condition that a solution of a program should satisfy. However, approximation constraints may be left unsatisfied, e.g. due to lack of computation time. This induces a partial order on such “approximate” solutions, where better approximations satisfy more constraints. Combined with the results from Section 2, it then becomes possible to devise an incremental extended answer set solver that combines both the grounding process and the computation, i.e. a “ground when needed” computation¹¹.

Definition 4 A simple approximation logic program (SALP) is a tuple (P, C) , where P is a SLP and C is a set of **approximation constraints**, i.e. rules of the form $\leftarrow \alpha$, with α a finite set of literals¹².

For an interpretation I of P , an approximation constraint $c \in C$ is said to be **satisfied**, denoted $I \models c$, if $\alpha \not\subseteq I$; otherwise it is said to be **violated**, denoted $I \not\models c$. We use I_v^C to denote the set of approximation constraints in C that are violated w.r.t. I , i.e. $I_v^C = \{c \in C \mid I \not\models c\}$.

Extended answer sets of P are **approximate extended answer sets** of (P, C) . An approximate extended answer set S of (P, C) is an **extended answer set** of (P, C) iff $S_v^C = \emptyset$, i.e. all approximation constraints are satisfied w.r.t. S .

In the rest of this paper, we will use $\mathcal{E}\mathcal{S}_n(P, C)$, with $0 \leq n \leq |C|$, to denote the set containing the approximate extended answer

sets of (P, C) that violate less than $n + 1$ approximation constraints in C , i.e. $\mathcal{E}\mathcal{S}_n(P, C) = \{S \in \mathcal{E}\mathcal{S}(P) \mid |S_v^C| \leq n\}$. Clearly, we have that $\mathcal{E}\mathcal{S}_0(P, C)$ corresponds to the extended answer sets of (P, C) ; and $\mathcal{E}\mathcal{S}_0(P, C) \subseteq \mathcal{E}\mathcal{S}_1(P, C) \subseteq \dots \subseteq \mathcal{E}\mathcal{S}_{|C|}(P, C) = \mathcal{E}\mathcal{S}(P)$.

Example 5 Consider the following SALP (P, C) with P

$$\begin{array}{lll} \neg a \leftarrow & \neg b \leftarrow & \neg c \leftarrow \\ b \leftarrow \neg a & a \leftarrow \neg b & c \leftarrow a \end{array}$$

and the following approximation constraints C

$$\leftarrow \neg a, \neg b \quad \leftarrow a \quad \leftarrow c$$

We have four extended answer sets for P , i.e. $\mathcal{E}\mathcal{S}(P) = \{S_1, S_2, S_3, S_4\}$, with $S_1 = \{\neg a, \neg b, \neg c\}$, $S_2 = \{\neg a, b, \neg c\}$, $S_3 = \{a, \neg b, \neg c\}$ and $S_4 = \{a, \neg b, c\}$. One can verify that $\mathcal{E}\mathcal{S}_2(P, Q) = \{S_1, S_2, S_3, S_4\}$, $\mathcal{E}\mathcal{S}_1(P, C) = \{S_1, S_2, S_3\}$ and $\mathcal{E}\mathcal{S}_0(P, C) = \{S_2\}$. Thus S_2 is the only extended answer set of the above SALP (P, C) .

In general, we may consider a partial order \preceq on extended answer set approximations which is such that better approximations are smaller. i.e. $S_1 \prec S_2$ ¹³ iff S_1 is a better approximation of an extended answer set than is S_2 . It seems reasonable to demand that any such preference order satisfies $S_1 \preceq S_2 \Rightarrow |S_1^C| \leq |S_2^C|$, i.e. better approximations satisfy more approximation constraints; and that “full” extended answer sets correspond to \prec -minimal approximations.

Two obvious (others are possible) candidates that satisfy this requirement are:

- cardinality, i.e. $S_1 \preceq S_2$ iff $|S_1^C| \leq |S_2^C|$; and
- subset, i.e. $S_1 \preceq S_2$ iff $S_1^C \subseteq S_2^C$

which are also used in other problem areas, such as diagnostic systems [13], that deal with preference on candidate solutions.

Note that, if $\mathcal{E}\mathcal{S}_0(P, C) = \emptyset$, i.e. there are no extended answer sets, and depending on the preference relation used, a \prec -minimal approximate extended answer set S may be useful, e.g. to provide insight, via the set of violated constraints S_v^C , about possible problems with the program. In this way a semantical debugging scheme for (extended) answer set programming can be defined, a concept which has already been explored in the context of prolog [3, 7, 12], but still needs better exploration in the former setting.

The following result sheds some light on the complexity of reasoning with approximation constraints. As $\mathcal{E}\mathcal{S}_{|C|}(P, C) = \mathcal{E}\mathcal{S}(P)$, we will only consider the cases $\mathcal{E}\mathcal{S}_i(P, C)$ with $0 \leq i < |C|$ (note that this implies that $|C| \geq 1$).

Theorem 3 Let (P, C) be a grounded SALP and take i such that $0 \leq i < |C|$. Deciding whether there exists an approximate extended answer set $S \in \mathcal{E}\mathcal{S}_i(P, C)$ is \mathcal{NP} -complete.

Again an exponential factor [6] or, in case of small bounded predicate arities, an additional level of the polynomial hierarchy [8] has to be added in the case of non-ground SALPs, i.e. either NEXPTIME-complete or Σ_2^P -complete respectively.

The above theorem implies that the semantics already gets its full computational complexity with a single constraint, which is not a surprise as we can replace each constraint $\leftarrow \alpha \in C$ with a rule $\text{inconsistent} \leftarrow \alpha$ in P and take $C = \{\leftarrow \text{inconsistent}\}$. However, the above complexity result is a worst-case scenario (just as

¹¹ Note that current answer set solvers lack this support for “ground when needed” (see Section 4) and compute first the complete grounding, instead.

¹² Note that we allow non-ground constraints. However, grounding a SALP can be done by computing $gr(P \cup C)$ and splitting up the result in two sets, i.e. the ground rules and the ground constraints. Again, we assume grounded SALPs in the rest of the paper.

¹³ As usual, $S_1 \prec S_2$ iff $S_1 \preceq S_2$ and not $S_2 \preceq S_1$.

with grounding), i.e. in an implementation we may expect, in the average case, that more constraints require more computation time.

Finally, the framework developed above can be used to implement an incremental extended answer set solver, i.e. a solver that produces an approximate solution and then, as long as resources permit, produces successively better approximations. Given enough resources, this approach will eventually result in an extended answer set, i.e. an approximation that satisfies all constraints.

Intuitively, such an implementation will start by computing an extended answer set S along the lines of the immediate consequence operator presented in Section 2. Afterwards, a backtracking algorithm will try to improve the approximation by selecting a constraint to satisfy, while maintaining the partial ordering defined on approximations.

4 Classical Answer Set Approximation

Theorem 3 indicates that the semantics presented in Section 3 is computationally equivalent to the classical answer set semantics [9] for non-disjunctive programs containing negation as failure in the body of rules. Here we show how the classical answer set semantics can be effectively translated to approximate extended answer sets, thus making it possible to compute classical answer sets via successive approximations.

We briefly introduce the answer set semantics for semi-negative programs. Such programs are build using atoms and naf-atoms, i.e. for an atom a , we use $\text{not } a$ to denote its negation-as-failure (naf) version, which intuitively means that $\text{not } a$ is true when a is not true. A semi-negative logic program P is a countable set of rules of the form $a \leftarrow \beta$ where a is an atom and β is a finite set of (naf-)atoms. An interpretation I for a semi-negative program P is a subset $I \subseteq \mathcal{B}_P$. An atom a is satisfied w.r.t. I , denoted $I \models a$ if $a \in I$; while a naf-atom $\text{not } a$ is satisfied w.r.t. I , i.e. $I \models \text{not } a$, when $I \not\models a$. The answer set semantics for semi-negative programs is defined in two steps.

First, consider programs without naf-atoms. For such a naf-free program P , an interpretation I is an answer set iff $P^* = I$. Next, for semi-negative programs, we use the Gelfond-Lifschitz transformation [9]. The idea behind this transformation is, for a given program P and a candidate solution S , to remove all naf constructs w.r.t. S and then check if the candidate solution is supported by the reduct program. Formally, for a semi-negative program P and a candidate solution S , the GL-reduct of P w.r.t. S , denoted P^S , is obtained from P by (a) removing each rule $a \leftarrow \beta$ with $\text{not } b \in \beta$ and $b \in S$, and (b) remove all naf-atoms from the remaining rules. Clearly, the program P^S is free from negation as failure. An interpretation S is an answer set of a semi-negative program P iff S is an answer set of P^S , i.e. $P^{S^*} = S$. We use $\mathcal{AS}(P)$ to denote the set containing all answer sets of P .

Example 6 Consider the following semi-negative program.

$$\begin{array}{ll} a \leftarrow \text{not } b & b \leftarrow \text{not } a \\ c \leftarrow a & d \leftarrow b \end{array}$$

Intuitively, the above program represents an exclusive choice between a or b and depending on this choice, the program derives an additional atom c or d resp. Indeed, one can verify that the above program has two answer sets, i.e. $S_1 = \{a, c\}$ and $S_2 = \{b, d\}$. E.g., the reduct P^{S_1} contains the rules $\{a \leftarrow \text{not } c \leftarrow a \quad d \leftarrow b\}$, for which $P^{S_1*} = \{a, c\} = S_1$.

We construct, for a semi-negative logic program P , a SALP $L(P) = (Q, C)$ such that the extended answer sets of $L(P)$, i.e. elements of $\mathcal{ES}_0(Q, C)$, are in one-to-one correspondence with the answer sets of P .

In the following definition we use α' to denote the set of literals obtained from a set α where each naf-atom $\text{not } a$ from α is replaced by $\neg a$. The notation is further extended to rules and programs.

Definition 5 Let P be a semi-negative program. The SALP $L(P) = (Q, C)$ is defined by $Q = P' \cup \{\neg a \leftarrow \text{not } a \mid a \in \mathcal{B}_P\}$ and $C = \{\leftarrow \beta', \neg a \mid a \leftarrow \beta \in P\}$.

Intuitively, we simulate negation as failure in two steps. First, we introduce negation as failure explicitly in Q using classical negation by introducing a fact $\neg a$ for each atom $a \in \mathcal{B}_P$. Secondly, we assert approximation constraints in C to enforce the satisfaction of the original rules in the program P : a rule $a \leftarrow \beta' \in Q$ (i.e. $a \leftarrow \beta \in P$) is only satisfied by an interpretation iff the approximation constraint $\leftarrow \beta', \neg a \in C$ is satisfied w.r.t. that interpretation.

Example 7 Reconsider the program from Example 6. Its SALP version $L(P) = (Q, C)$ is defined by the SLP Q

$$\begin{array}{llll} \neg a \leftarrow & \neg b \leftarrow & \neg c \leftarrow & \neg d \leftarrow \\ a \leftarrow \neg b & b \leftarrow \neg a & c \leftarrow a & d \leftarrow b \end{array}$$

and the set of constraints C

$$\begin{array}{llll} \leftarrow \neg b, \neg a & \leftarrow \neg a, \neg b & \leftarrow a, \neg c & \leftarrow b, \neg d \end{array}$$

Consider the following extended answer sets of Q , i.e. $S'_1 = \{a, \neg b, c, \neg d\}$, $S'_2 = \{\neg a, b, \neg c, \neg d\}$ and $S'_3 = \{a, \neg b, \neg c, \neg d\}$. One can check that $\mathcal{ES}_0(Q, C) = \{S'_1, S'_2\}$ and clearly $S_1 = S'_1 \cap \mathcal{B}_P$ and $S_2 = S'_2 \cap \mathcal{B}_P$. On the other hand, $S'_3 \in \mathcal{ES}_1(Q, C)$, as $c \leftarrow a \in C$ is violated. However, one can easily see that $S_3 = S'_3 \cap \mathcal{B}_P$ is an answer set of the program $P \setminus \{c \leftarrow a\}$.

The behavior outlined in the previous example is confirmed in general by the following theorem.

Theorem 4 Let P be a semi-negative logic program and consider $L(P) = (Q, C)$ as defined in Definition 5. Then, for $S \subseteq \mathcal{B}_P$,

$$S \cup \{\neg l \mid l \in \mathcal{B}_P \setminus S\} \in \mathcal{ES}_0(Q, C) \iff S \in \mathcal{AS}(P)$$

i.e. the answer sets of P are in one-to-one correspondence with the extended answer sets of $L(P)$.

Further, for i , with $0 < i \leq |C|$, we have, with

$S \cup \{\neg l \mid l \in \mathcal{B}_P \setminus S\} \in (\mathcal{ES}_i(Q, C) \setminus \mathcal{ES}_{i-1}(Q, C)) \iff S \in \mathcal{AS}(P \setminus S_v^C)$, where, abusing notation, $P \setminus S_v^C$ denotes the subset of rules in P for which the corresponding approximation constraint in C is satisfied w.r.t. S .

The above result shows that the approximation framework for extended answer sets is able to handle the classical answer set semantics in the sense that, each time a better approximation for $L(P)$ is computed, it is also an answer set for a larger subset of the program P . Interestingly, the algorithm does not rely on negation as failure, but uses only classical negation and the ability to leave rules unsatisfied, i.e. defeated.

Finally, the above results may help to resolve another issue regarding current answer set solvers such as DLV [11] and SMODELS [14]. Indeed, these solvers require that a program first be completely grounded before the actual answer set computation starts. With approximations based on extended answer sets, this is probably not necessary, since the computation of an extended answer set may operate on a “ground (a rule) as needed” basis (Section 2).

5 Related Work

The idea of an approximation theory in logic and logic programming is not new. A good example of this is the “anytime” family of reasoners for propositional logic [4]. Intuitively, such a system consists of a sequence $\vdash_0, \vdash_1, \dots, \vdash_c$ of inference relations such that each \vdash_i is at least as good¹⁴ as \vdash_{i-1} and there exists a complete inference relation \vdash_c for the problem. Using such an anytime reasoner, one starts the computation with the inference relation \vdash_0 and as long as there is computation time left, the inference relations $\vdash_1, \vdash_2, \dots$ are applied to obtain better approximations. When the complete reasoner \vdash_c is reached during the computation, one had enough resources available to compute a completely correct solution.

This idea is e.g. applied in [5] to obtain an anytime reasoner for Boolean Constraint Propagation, which is used in [18] to obtain an approximation theory for diagnostic reasoning.

In [1] a first attempt is made to devise a framework for interactive answer set programming. The idea is to find which part of a program needs to be recomputed in order to find a new answer set in case a single rule is added to the program. Using this framework, one can obtain a similar approximative version of answer set programming as the one presented in Section 4 by starting the computation with a single rule and then subsequently adding new rules. However, the framework presented in this paper is more suitable in the sense that we can start the computation with the whole program, not just a single rule, and obtain a first approximation that in the average case will satisfy more than one rule in the program under consideration.

Weak constraints were introduced in [2] as a relaxation of the concept of a constraint. Intuitively, a weak constraint is allowed to be violated, but only as a last resort, meaning that one tries to minimize the set of violated constraints (e.g. using either subset minimality or cardinality minimality). The main difference of this approach with ours, is that we use approximation constraints to find better approximations, as long as we have resources available, of a final solution that satisfies all approximation constraints (and which will be reached if we have enough time); while the weak constraints in [2] serve to differentiate between the answer sets of the program without weak constraints, by posing additional conditions on those answer sets that do not necessarily have to be satisfied. Also note that the weak constraints from [2] have a much higher complexity than the framework presented in this paper, i.e. Δ_2^P for non-disjunctive programs, or the second level of the polynomial hierarchy.

6 Conclusions and Directions for Further Research

We presented a first attempt at an approximation theory, by means of approximation constraints, for the extended answer set semantics of programs containing only classical negation. Further, we showed how classical answer sets can be approximated using the proposed framework.

In future work, we will implement an approximate extended answer set solver that “grounds when needed”. In our implementation, we will try to incorporate some of the ideas developed in the Platypus system [10], i.e. a system to compute answer sets in a distributed environment, to obtain an implementation that can be used in a distributed setting. One topic for further research in this context is e.g. how on the fly rule instantiations (which result from the “ground when needed” process) have to be propagated to the other participants in the distributed computation process.

Furthermore, we also plan to look into the related problems of profiling (finding out which parts of a program are hard to compute) and debugging (finding out which parts of a program are wrong) for (extended) answer set programming. Finally, the present approach could be generalized to support user-defined preference relations (“hints”) on approximation constraints which could be used to influence the approximation process.

REFERENCES

- [1] M. Brain, R. Watson, and M. De Vos, ‘An interactive approach to answer set programming’, in *Proc. of the 3rd Intl. Workshop on ASP: Advances in Theory and Implementation (ASP05)*, volume 142 of *CEUR Workshop Proceedings*, pp. 190–202, (2005).
- [2] Francesco Buccafurri, Nicola Leone, and Pasquale Rullo, ‘Strong and weak constraints in disjunctive datalog’, in *Proc. of the 4th Intl. Conference on Logic Programming (L��NMR ’97)*, pp. 2–17, (1997).
- [3] M. Calejo and L.M. Pereira, ‘Declarative source debugging’, in *Proc. of the 5th Portuguese Conf. on AI (EPIA91)*, volume 541 of *LNCS*, pp. 237–249. Springer, (1991).
- [4] Mukesh Dalal, ‘Anytime families of tractable propositional reasoners’, in *Proc. of the 4th Intl. Symp. on Artificial Intelligence and Mathematics (AI/MATH96)*, pp. 42–45, (1996).
- [5] Mukesh Dalal, ‘Semantics of an anytime family of reasoners’, in *Proceedings of the 12th European Conference on Artificial Intelligence*, pp. 360–364. John Wiley and Sons, (1996).
- [6] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, ‘Complexity and Expressive Power of Logic Programming’, *ACM Computing Surveys*, **33**(3), 374–425, (2001).
- [7] Mireille Ducassé, ‘A pragmatic survey of automated debugging’, in *Proc. of the Intl. Workshop on Automated and Algorithmic Debugging (AADEBUG93)*, volume 749 of *LNCS*, pp. 1–15. Springer, (1993).
- [8] Thomas Eiter, Wolfgang Faber, Michael Fink, Gerald Pfeifer, and Stefan Woltran, ‘Complexity of model checking and bounded predicate arities for non-ground answer set programming’, in *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pp. 377–387. AAAI Press, (2004).
- [9] Michael Gelfond and Vladimir Lifschitz, ‘The stable model semantics for logic programming’, in *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pp. 1070–1080, Seattle, Washington, (August 1988). The MIT Press.
- [10] Jean Gressmann, Tomi Janhunen, Robert E. Mercer, Torsten Schaub, Sven Thiele, and Richard Tichy, ‘Platypus: A platform for distributed answer set solving’, in *Logic Programming and Nonmonotonic Reasoning: 8th International Conference (LPNMR 2005)*, volume 3662 of *LNCS*, pp. 227–239. Springer, (2005).
- [11] N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell’Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, C. Koch, S. Perri, and A. Polleres, ‘The dlv system’, in *Proc. of the Eur. Conf. on Logics in AI (JELIA2002)*, volume 2424 of *LNCS*, pp. 537–540. Springer, (2002).
- [12] Lee Naish, ‘A declarative debugging scheme’, *Journal of Functional and Logic Programming*, **1997**(3), (1997).
- [13] Raymond Reiter, ‘A theory of diagnosis from first principles’, *Artificial Intelligence*, **32**(1), 57–95, (1987).
- [14] Patrik Simons, Ilkka Niemelä, and Timo Soininen, ‘Extending and implementing the stable model semantics’, *Artificial Intelligence*, **138**(1–2), 181–234, (2002).
- [15] M. H. van Emden and R. A. Kowalski, ‘The semantics of predicate logic as a programming language’, *Journal of the Association for Computing Machinery*, **23**(4), 733–742, (1976).
- [16] Davy Van Nieuwenborgh, Stijn Heymans, and Dirk Vermeir, ‘Approximating extended answer sets’, Technical report, Vrije Universiteit Brussel, Dept. of Computer Science, 2006, <http://tinf2.vub.ac.be/dvnieuwe/ecai2006technical.ps>.
- [17] Davy Van Nieuwenborgh and Dirk Vermeir, ‘Preferred answer sets for ordered logic programs’, in *Proc. of the Eur. Conf. on Logics in Artificial Intell. (JELIA2002)*, volume 2424 of *LNCS*, pp. 432–443. Springer, (2002).
- [18] A. Verberne, F. van Harmelen, and A. ten Teije, ‘Anytime diagnostic reasoning using approximate boolean constraint propagation’, in *Proc. of the 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pp. 323–332. Kaufman, (2000).

¹⁴ Note that each \vdash_i has to be sound and tractable, but not necessarily complete.

An Axiomatic Approach to Qualitative Decision Theory with Binary Possibilistic Utility

Paul Weng¹

Abstract. Binary possibilistic utility unifies two previously proposed qualitative decision models: optimistic and pessimistic utilities. All these decision models have been axiomatized in a von Neumann-Morgenstern setting. These axiomatizations have shown the formal similarity of these qualitative utilities and expected utility. Unfortunately in this framework, the representation of uncertainty has to be clearly assumed to be given. In a more general setting, without this restriction, optimistic and pessimistic utilities have been axiomatized à la Savage. This paper proposes a study of the axioms of binary possibilistic utility in a Savagean framework.

1 INTRODUCTION

Decision making under uncertainty is an important topic in AI research, whether in the process of building autonomous agents or in modeling the behavior of other agents (human or not) in the course of a multi-agent interaction, since any artificial agent can be viewed as an automated decision maker [21]. Decision theory has been first studied in economics, which explains why quantitative decision models are the most common ones and have been the most investigated. Among those is the classic model of expected utility (EU). The axiomatic works of von Neumann and Morgenstern (vNM) [26] and Savage [22] highlighted the properties of this criterion and described the situations where it should be employed. This model is known to make some strong hypotheses: uncertainty has to be represented by probability and a numeric value (utility) is assigned to each consequence of an action. These assumptions are not suitable to all kinds of problems. Indeed, in the situations where data are imprecise and/or scarce, the use of probability is sometimes debatable. Even when probability is appropriate, the assessment of the parameters of the model (probabilities and utilities) is a difficult and information demanding task.

For these reasons, it is of interest to study other decision models. In the situations where probability is not suitable, many alternatives have been proposed: theory of evidence [24], possibility theory [8], kappa rankings [25] or plausibility measures [13]. In each of these frameworks, decision models have been proposed [18, 9, 14, 4].

In AI, with its long standing tradition of symbolic problem modeling and solving, qualitative decision models ([2], [3], [19],...) have received much attention. They are very appealing as they are not information demanding (only qualitative information is required) and lead to faster computations. In this paper, we restrict ourself to qualitative decision models based on possibility theory and more specifically to binary possibilistic utility (PU) that was introduced by Giang and Shenoy [15]. This qualitative model has many interesting properties. It unifies two previously proposed qualitative utilities: optimistic

and pessimistic utilities [9, 6] and can model decision behavior that neither optimistic nor pessimistic utilities can model. Moreover it is dynamically consistent and therefore can be applied in sequential decision making, especially in Markov Decision Processes [20]. Finally it can be considered the qualitative counterpart of EU. Indeed, Weng [27] has shown that a same set of axioms (complete preorder, independence, continuity and non triviality) leads to either EU or PU depending on whether uncertainty is represented by respectively a probability or a possibility distribution.

Axiomatizing a decision criterion is essential as it reveals its properties and its limits. The axiomatization of EU highlighted its restrictions [1, 12]. Such work has to be done in the qualitative setting, which encompasses logic-formalized decision problems [9]. Axiomatization can be lead in two different frameworks: in a vNM setting where the nature of uncertainty is assumed to be given here of possibilistic nature or in a Savagean setting where no such prior assumption is made. The axiomatic studies of this criterion have all been lead in a vNM framework [15, 16]. Dubois et al. [10] axiomatized optimistic and pessimistic utilities in a Savagean context. These works give new insights about the two qualitative utilities in a very general framework. Following their works, we propose here to give a Savagean axiomatization of PU. Interestingly, such works allow the nature of uncertainty as perceived by the decision maker to be revealed: possibility for optimistic utility, necessity for pessimistic utility and as shown in this paper a pair consisting of possibility and necessity for PU. Consequently, this gives some justifications for the use of non probabilistic uncertainty representation.

This paper is organized as follows. In Section 2, we recall all the necessary notions for the presentation of our results. Due to lack of space, we do not present the Savagean axiomatics for EU and its comparison with that of qualitative utilities. We refer the interested reader to [10]. Then, in Section 3, we present some properties of PU for a better understanding of this decision model. In Section 4, we present our axiomatization. Finally we conclude in Section 5.

2 POSSIBILISTIC UTILITIES

2.1 Some Definitions and Notations

In decision under uncertainty, a decision maker has to choose an action, named act, whose consequence is not known precisely. The decision problem can be described by the following elements:

- S a finite set of states.
- $X = \{x_1, \dots, x_n\}$ a finite set of consequences. Elements 0_X and 1_X denote respectively the worst and the best consequences.
- \mathcal{F} a finite set of acts, i.e. functions from the set of states to the set of consequences ($\mathcal{F} = X^S$).

¹ LIP6, France, email: paul.weng@lip6.fr

A state (or state of nature) describes a possible configuration of the environment. The decision maker has to choose an act, which is a function from the set of states to the set of consequences. The decision maker's preference relation (which is simply a binary relation) over acts is denoted by $\succsim_{\mathcal{F}}$, which reads "at least as good as".

Uncertainty follows from the fact that the decision maker does not know precisely in which state he is. Uncertainty is then defined as uncertainty about the actual state of nature. Events are simply sets of states. The set of all events is denoted 2^S . For two events A and B , $A \succsim_S B$ means that A is judged at least as plausible than B by the decision maker. The complement of an event $A \in 2^S$ is denoted \overline{A} .

For two acts f and g , an event A , fAg denotes the act consisting in doing f if event A and doing g otherwise, i.e. $\forall s \in A, fAg(s) = f(s)$ and $\forall s \in \overline{A}, fAg(s) = g(s)$. The constant act equal to $x \in X$ everywhere is denoted \mathbf{x} . Relation $\succsim_{\mathcal{F}}$ restricted to constant acts induces a preference relation over consequences, denoted \succsim_X .

For any relation \succsim , \succ and \sim denote respectively the asymmetric and the symmetric part of \succsim .

A finite qualitative scale L on which consequences and uncertainty can be measured is defined. Its lowest and its greatest elements are written respectively by 0_L and 1_L . The order on L is denoted by \geq_L . Operators max and min on scale L are respectively written \vee and \wedge . The order reversing involution in L is denoted by n , i.e. $n(0_L) = 1_L, n(1_L) = 0_L$ and $\forall \lambda, \lambda' \in L, \lambda \geq_L \lambda' \Rightarrow n(\lambda) \leq_L n(\lambda')$. By abuse of notation, in the paper, n will denote any such one-to-one mapping that reverses a preordered scale.

A *basic utility assignment* is a mapping that associates a value in a scale to each consequence in X . A *capacity* $\sigma : 2^S \rightarrow L$ is a very general measure of uncertainty that satisfies:

- $\sigma(\emptyset) = 0_L$
- $\sigma(S) = 1_L$
- $\forall A, B \in 2^S, A \subseteq B \Rightarrow \sigma(A) \leq_L \sigma(B)$

As examples, probability, possibility, belief functions, kappa rankings and plausibility measures are all capacities. We recall the definitions of a possibility distribution. A capacity is a possibility distribution (denoted π) iff it additionally satisfies $\forall A \in 2^S, \pi(A) = \bigvee_{s \in A} \pi(s)$. Distribution π can be viewed as an encoding of the ranking of the states in terms of plausibility. It can be interpreted as follows: $\pi(s) = 0_L$ means that state s is impossible, $\pi(s) = 1_L$ means that s is totally possible, when $0_L <_L \pi(s) <_L 1_L$, s is only possible to some extent, i.e. there are states more possible than s . We assume that there exists at least one state of possibility 1_L and of course, there could be several ones.

We introduce a few axioms on the preorder over events.

A1 Relation \succsim_S over 2^S is a total preorder

A2 $S \succsim_S \emptyset$

A3 $\forall A \in 2^S, A \succsim_S \emptyset$

These axioms corresponds to the properties of a capacity. The following theorem can then be stated [11]:

Theorem 1. If \succsim_S , a preorder over states, satisfies axioms **A1**, **A2** and **A3** then it exists a finite qualitative scale (L, \geq_L) and a capacity $\sigma : 2^S \rightarrow L$ such that $A \succsim_S B \Leftrightarrow \sigma(A) \geq_L \sigma(B)$.

Adding the following axiom, the capacity becomes a possibility distribution [5].

POS $\forall A, B, C \in 2^S, A \succsim_S B \Rightarrow A \cup C \succsim_S B \cup C$.

Dually the following axiom characterizes a necessity distribution.

NES $\forall A, B, C \in 2^S, A \succsim_S B \Rightarrow A \cap C \succsim_S B \cap C$.

2.2 Sugeno Integral

Before introducing the qualitative utilities, we present the Sugeno integral, which is a very general decision criterion. It can be seen as the qualitative counterpart of Choquet expected utility (CEU) [17, 23]. Here, the basic utility assignment is written $\mu : X \rightarrow L$.

The utility of an act defined as a Sugeno integral is defined:

$$U_S(f) = \bigvee_{x \in X} (\mu(x) \wedge \sigma(F_x)) \quad (1)$$

where $F_x = \{s \in S : f(s) \succsim_X x\}$.

To understand this formula, Dubois and Prade [7] showed that $U_S(f)$ is the median of the utilities and uncertainty values in the following set: $\{\mu(x) : x \in X\} \cup \{\sigma(F_x) : x \in X, x \succsim_X 0_X\}$. This highlights the similarity with the quantitative models (CEU or EU), which computes a weighted average.

The axioms used for the axiomatization of Sugeno integrals are:

Sav1 Preference relation $\succsim_{\mathcal{F}}$ over \mathcal{F} is a total preorder.

WS3 $\forall x, y \in X, x \succsim_{\mathcal{F}} y \Rightarrow \forall A \in 2^S, \forall h \in \mathcal{F}, xAh \succsim_{\mathcal{F}} yAh$

Sav5 $\exists x, x' \in X, x \succsim_{\mathcal{F}} x'$

RCD $\forall f, g \in \mathcal{F}, \forall x \in X, g \succsim_{\mathcal{F}} f$ and $x \succsim_{\mathcal{F}} f \Rightarrow g \wedge x \succsim_{\mathcal{F}} f$

RDD $\forall f, g \in \mathcal{F}, \forall x \in X, f \succsim_{\mathcal{F}} g$ and $f \succsim_{\mathcal{F}} x \Rightarrow f \succsim_{\mathcal{F}} g \vee x$

Axiom **Sav1** states that any two acts can be compared and the preference relation is transitive. Axiom **WS3** declares that when there is a preference between two consequences, this preference can not reverse if considering two acts giving these consequences on a certain event and the same consequences on the complementary event. Axiom **Sav5** is enforced in order to avoid triviality. Axiom **RCD** states that lowering the consequences of an act f , which is preferred to an act g , to a constant consequence that is also preferred to g still yields a better act than g . Finally axiom **RDD** is the dual of **RCD**. It says that an act f is preferred to an act g even if the worst consequences of g are improved to a constant consequence that is less preferred than f . By way of information, EU verifies neither **RCD** nor **RDD** [10].

Sugeno integrals [10] are characterized by:

Theorem 2. If $\succsim_{\mathcal{F}}$, a preorder over \mathcal{F} , satisfies **Sav1**, **WS3**, **Sav5**, **RCD** and **RDD** then there is a finite qualitative scale (L, \geq_L) , a basic utility assignment $\mu : X \rightarrow L$ and a capacity $\sigma : 2^S \rightarrow L$ such that $f \succsim_{\mathcal{F}} f' \Leftrightarrow U_S(f) \geq_L U_S(f')$.

2.3 Optimistic and Pessimistic Utilities

We now present optimistic and pessimistic utilities, axiomatized in a von Neumann-Morgenstern [6] and a Savagean [10] settings. In this framework, the basic utility assignment is denoted $v : X \rightarrow L$.

Then optimistic and pessimistic utilities are functions from \mathcal{F} to L . Optimistic utility writes:

$$U^+(f) = \bigvee_{s \in S} (\pi(s) \wedge v(f(s))) \quad (2)$$

and pessimistic utility writes:

$$U^-(f) = \bigwedge_{s \in S} (n(\pi(s)) \vee v(f(s))) \quad (3)$$

Optimistic utility is a Sugeno integral with the capacity taken as a possibility distribution. It is a generalization of the maximax criterion axiomatized by Brafman and Tennenholz [3]. And pessimistic utility is a Sugeno integral when the capacity is a necessity distribution. It is a generalization of the maximin criterion [3].

To enforce this utility, axiom **RDD** needs to be replaced by a stronger axiom:

$$\text{OPT } \forall f, g \in \mathcal{F}, \forall A \in 2^S, f \succ_{\mathcal{F}} f A g \Rightarrow g A f \sim_{\mathcal{F}} f$$

This axiom says that if in a certain event, changing the consequences of an act yields a strictly less preferred one then in the complementary event, changing its consequences cannot yield a strictly less preferred act. It reveals a particular property of optimistic utility. If an act can be strictly downgraded in a certain event, this event is judged plausible enough to attract all the attention of the decision maker and what happens on the complementary event is not relevant.

The representation theorem for the optimistic utility [10] states:

Theorem 3. *If $\sim_{\mathcal{F}}$, a preorder over \mathcal{F} , satisfies **Sav1**, **WS3**, **Sav5**, **RCD** and **OPT** then there is a finite qualitative scale (L, \geq_L) , a basic utility assignment $v : X \rightarrow L$ and a possibility distribution $\pi : 2^S \rightarrow L$ such that $f \sim_{\mathcal{F}} f' \Leftrightarrow U^+(f) \geq_L U^+(f')$.*

For the pessimistic utility, axiom **RCD** is to be replaced by:

$$\text{PES } \forall f, g \in \mathcal{F}, \forall A \in 2^S, f A g \succ_{\mathcal{F}} f \Rightarrow f \sim_{\mathcal{F}} g A f$$

This axiom, which is the dual of **OPT** states that if an act is strictly improved in a certain event then it can not be strictly improved in the complementary event.

And the pessimistic utility [10] is characterized by:

Theorem 4. *If $\sim_{\mathcal{F}}$, a preorder over \mathcal{F} , satisfies **Sav1**, **WS3**, **Sav5**, **RDD** and **PES** then there is a finite qualitative scale (L, \geq_L) , a basic utility assignment $v : X \rightarrow L$ and a possibility distribution $\pi : 2^S \rightarrow L$ such that $f \sim_{\mathcal{F}} f' \Leftrightarrow U^-(f) \geq_L U^-(f')$.*

2.4 Binary Possibilistic Utility

Binary possibilistic utility (PU) introduced by Giang and Shenoy [15] unifies the two previous qualitative decision models. It takes values in a particular scale L_2 , which is a binary scale defined by the following set of pairs:

$$L_2 = \{(\lambda, \mu) : \lambda, \mu \in L, \lambda \vee \mu = 1_L\}.$$

Due to constraint $\lambda \vee \mu = 1_L$, scale L_2 is in fact monodimensional (see fig. 1). A pair in L_2 can be interpreted as a binary possibility

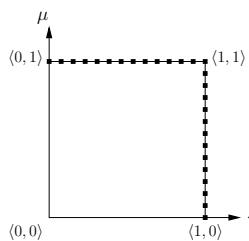


Figure 1. Binary utility scale L_2 .

distribution over two consequences : the best and the worst ones. The first element of the pair would be the possibility to get the best consequence and the second would be the possibility to get the worst consequence. Then a natural order relation² can be defined on L_2 :

$$(\lambda, \mu) \geq_{L_2} (\lambda', \mu') \iff (\lambda \geq_L \lambda' \text{ and } \mu \leq_L \mu').$$

² This order is expressed in a simpler form than in [15, 16]. It can be easily checked that the two forms are equivalent.

Indeed, the decision maker naturally prefers the highest possibility of getting the best outcome and the lowest possibility of getting the worst one.

Here, the basic utility assignment is denoted $u : X \rightarrow L_2$. Remark that it takes values in the binary scale. Uncertainty represented by a possibility distribution is still measured on L .

Operator \vee is extended as an operator on $L_2 \times L_2$ as follows:

$$\langle \lambda, \mu \rangle \vee \langle \lambda', \mu' \rangle = \langle \lambda \vee \lambda', \mu \vee \mu' \rangle.$$

Remark that this operator is not operator max on L_2 , which would be computed as $\max(\langle \lambda, \mu \rangle, \langle \lambda', \mu' \rangle) = \langle \lambda \vee \lambda', \mu \wedge \mu' \rangle$.

Operator \wedge is extended as an operator on $L \times L_2$:

$$\lambda' \wedge \langle \lambda, \mu \rangle = \langle \lambda' \wedge \lambda, \lambda' \wedge \mu \rangle.$$

In the same way as operator max, operator min on L_2 is given by $\min(\langle \lambda, \mu \rangle, \langle \lambda', \mu' \rangle) = \langle \lambda \wedge \lambda', \mu \vee \mu' \rangle$.

PU is then defined as a function from \mathcal{F} to L_2 :

$$PU(f) = \bigvee_{s \in S} (\pi(s) \wedge u(f(s))). \quad (4)$$

As noted in [15], when the basic utility has the following restricted form: $\forall x \in X, \exists \lambda \in L, u(x) = \langle \lambda, 1_X \rangle$ then PU is an optimistic utility. Dually when we have $\forall x \in X, \exists \mu \in L, u(x) = \langle 1_X, \mu \rangle$, PU is a pessimistic utility. Finally we remark that compared to PU, optimistic and pessimistic utilities exploit only half of scale L_2 .

3 STUDY OF THE PU MODEL

3.1 Properties of PU

We assume here that no two consequences in X are equivalent. This is not a restrictive condition as if it were not the case, one could work on the equivalence classes of X instead of X . This condition entails that X is a preordered scale. Let $n \circ f$ be the act that is defined by $\forall s \in S, (n \circ f)(s) = n(f(s))$. Then the *dual* of a preference relation $\sim_{\mathcal{F}}$ over \mathcal{F} , denoted $\sim_{\mathcal{F}}^T$, is defined by $f \sim_{\mathcal{F}}^T g \Leftrightarrow n \circ g \sim_{\mathcal{F}} n \circ f$. Obviously the dual of the dual of a preference relation is the relation itself. By extension, we say that a decision model is the dual of another one if the preference relation defined by the former decision model is the dual of the relation defined by the latter. Pessimistic utility is the dual of optimistic utility.

A preference relation is said to be *autodual* if and only if $\sim_{\mathcal{F}}^T = \sim_{\mathcal{F}}$. This means that for two acts f and g , $f \sim_{\mathcal{F}} g \Leftrightarrow n \circ g \sim_{\mathcal{F}} n \circ f$. Again, by extension, a decision model is said to be autodual if the induced preference relation is autodual. Intuitively, it means that the decision model when stating a preference between two acts looks at how well and how bad these two acts do. Any decision model that is not autodual does not use all the information at its disposal since it could be possible to exploit the dual of the decision model to yield a refined criterion. In this sense a rational decision maker would rather use an autodual decision model.

As the first and second part of PU are respectively of the form of an optimistic utility and a pessimistic one, PU induces an autodual preference relation over acts. EU is another example of such an autodual decision model. Once again, similarities between the two criteria can be emphasized.

When the basic utility assignment is valued in L_2 and uncertainty is measured by a possibility distribution, Eqs. 1 and 4 are equivalent.

Proposition 1. $PU(f) = U_S(f)$

Proof. We prove the result recursively on n the number of different consequences of f . Let $f = x_1 A_1 x_2 A_2 \dots x_n A_n$ where $x_1 \succ_X x_2 \succ_X \dots \succ_X x_n$. We want to prove that $PU(f) = U_S(f)$. Assume for $n = 2$ that $f = x A y$ with $x \succ_X y$. Then $PU(f) = \langle (u_1(x) \wedge \pi(A)) \vee (u_1(y) \wedge \pi(\bar{A})), (u_2(x) \wedge \pi(A)) \vee (u_2(y) \wedge \pi(\bar{A})) \rangle$.

$$\begin{aligned} U_S(f) &= \max(\langle u_1(y), u_2(y) \rangle, \\ &\quad \min(\langle u_1(x), u_2(x) \rangle, \langle \pi(A), \pi(\bar{A}) \rangle)) \\ &= \max(\langle u_1(y), u_2(y) \rangle, \langle u_1(x) \wedge \pi(A), u_2(x) \vee \pi(\bar{A}) \rangle) \\ &= \langle u_1(y) \vee (u_1(x) \wedge \pi(A)), u_2(y) \wedge (u_2(x) \vee \pi(\bar{A})) \rangle \\ &= \langle u_1(y) \vee (u_1(x) \wedge \pi(A)), u_2(x) \vee (u_2(y) \wedge \pi(\bar{A})) \rangle \end{aligned}$$

Now if $\pi(A) = 1_L$ then $PU(f) = \langle u_1(x) \vee (u_1(y) \wedge \pi(\bar{A})), u_2(x) \vee (u_2(y) \wedge \pi(\bar{A})) \rangle = \langle u_1(x), u_2(x) \vee (u_2(y) \wedge \pi(\bar{A})) \rangle$ and $U_S(f) = \langle u_1(y) \vee u_1(x), u_2(x) \vee (u_2(y) \wedge \pi(\bar{A})) \rangle = PU(f)$. In the same way, we can check that if $\pi(\bar{A}) = 1_L$ then $PU(f) = U_S(f)$. Now assume the property is true for n and prove that it is true for $n + 1$. We have $f = x_1 A_1 x_2 A_2 \dots x_{n+1} A_{n+1}$ where $x_1 \succ_X x_2 \succ_X \dots \succ_X x_{n+1}$. If $\pi(A_1) < 1_L$ apply the recursion assumption on $g = x_2 A_2 \dots x_{n+1} A_{n+1}$. Otherwise, apply the recursion assumption on $g = x_1 A_1 \dots x_n A_n \cup A_{n+1}$. We get $PU(g) = U_S(g)$. Then apply the case $n = 2$ to get the result. \square

The previous result is not surprising. Being the unification of optimistic and pessimistic utilities, which are two Sugeno integrals, PU is naturally also a Sugeno integral. Indeed PU satisfies all the required conditions of Theorem 2.

Proposition 2. PU satisfies **Sav1**, **WS3**, **Sav5**, **RCD** and **RDD**.

Consequently, in the PU model, uncertainty is not measured by a possibility distribution, as it could be at first thought, but by the pair of a possibility distribution and its associated necessity distribution. Indeed as noted in Proposition 1, uncertainty is measured by the pair $\langle \pi(A), \pi(\bar{A}) \rangle$. We recall that the necessity distribution that is associated to a possibility distribution is defined by $N(A) = n(\pi(\bar{A}))$.

Before going on with the properties of the underlying uncertainty measure, we introduce the axiom of weak independence. This is a weakening of the axiom of the sure thing principle used by Savage for EU.

$$\textbf{WI} \quad \forall f, g, h, h' \in \mathcal{F} \quad \left\{ \begin{array}{l} f A h \succ_{\mathcal{F}} g A h \\ \forall A \in 2^S \end{array} \right\} f A h' \succ_{\mathcal{F}} g A h'$$

This axiom is crucial in dynamic decision making and ensures that dynamic consistency holds. EU of course fulfills the sure thing principle and **WI**. Axiom **WI** is satisfied by PU guaranteeing that this criterion is dynamically consistent.

3.2 Properties of the Underlying Uncertainty Measure in PU

The properties of duality and autoduality can be also stated on the preorder over events. The *dual* of \succ_S is the relation \succ_S^T defined by $\forall A, B \in 2^S, A \succ_S^T B$ iff $\bar{B} \succ_S \bar{A}$. Preorder \succ_S is said to be *autodual* iff for two events A and B , $A \succ_S B \Leftrightarrow \bar{B} \succ_S \bar{A}$. This means that when A is more plausible than B , the complement of B is more plausible than that of A . Duality and autoduality can be naturally extended to uncertainty measures. As previously an uncertainty measure that is not autodual does not use all the information since it could be refined with its dual uncertainty measure. Therefore autodual uncertainty measure should be exploited when possible. A probability distribution is an example of such a measure.

The underlying uncertainty measure σ used in PU is said to be *autodual possibilistic*. It is defined by $\sigma : 2^S \rightarrow L_2, \forall A \in 2^S, \sigma(A) = \langle \pi(A), \pi(\bar{A}) \rangle$, which is obviously an autodual measure.

Some definitions are needed in order to characterize such capacities. For any event D , we write D^{\prec_S} for the set of events: $\{A \in 2^S : A \prec_S D\}$ and D^{\succ_S} for the set of events: $\{A \in 2^S : A \succ_S D\}$. They are respectively the set of events that are less plausible than D and its complement.

Now we introduce two axioms entailing that the capacity is an autodual possibilistic capacity.

AD^S Relation \succ_S over 2^S is autodual.

POS' $\exists D, S \succ_S D \succ_S \emptyset$ such that

$$\forall A, B, C \in D^{\prec_S}, A \succ_S B \Rightarrow A \cup C \succ_S B \cup C$$

Obviously axiom **AD^S** is required if we want an autodual capacity. Axiom **POS'** roughly states that the set of events can be divided into two parts such that on the less plausible side, \succ_S satisfies axiom **POS**. Together axioms **POS'** and **AD^S** give the dual property:

NES' $\exists D, S \succ_S D \succ_S \emptyset$ such that

$$\forall A, B, C \in D^{\succ_S}, A \succ_S B \Rightarrow A \cup C \succ_S B \cup C$$

Axioms **POS'** and **NES'** are of course fulfilled by autodual possibilistic capacities. If there are two states s, s' that have possibility 1_L then event D is such that its uncertainty value equals $\langle 1_L, 1_L \rangle$ (with $s \in D$ and $s' \in \bar{D}$). Otherwise, let s be the state of possibility 1_L and s' the most plausible state after s . Then D is such that $\sigma(D)$ equals $\langle 1_L, \pi(s') \rangle$.

Now autodual possibilistic capacities can be characterized:

Theorem 5. A capacity σ satisfies axioms **POS'** and **AD^S** if and only if it is an autodual possibilistic capacity.

Proof. First take D from axiom **POS'**. In the following, we write s for the set consisting only of state s . Remark that at least one event s is more plausible than \emptyset . Otherwise, if for all state s , $s \sim_S \emptyset$, by **POS'**, set S would be equivalent to \emptyset .

Choose a member in each equivalence class of S . We have $s_1 \succ_S s_2 \succ_S \dots \succ_S s_n$. Now each equivalence class of $2^S \cap D^{\prec_S}$ is equivalent to one of the s_i by **POS'**.

We have $s_1 \succ_S D$. Otherwise, by **POS'**, set S would be less plausible than D . By contradiction, assume that $s_1 \succ_S D$. Then s_1 is not in D by monotony of σ . Applying **AD^S** with **POS'**, we get **NES'**. Then with $A = s_1, B = D, C = D, A \succ_S B$ implies $A \cap C = \emptyset \succ_S D = B \cap C$, which is a contradiction. Then $s_1 \sim_S D$.

A possibility distribution $\pi : D^{\prec_S} \rightarrow L \setminus \{1_L\}$ can be constructed by **POS'**. Define $\sigma : D^{\prec_S} \rightarrow L_2$ by $\forall A \in D^{\prec_S}, \sigma(A) = \langle \pi(A), 1_L \rangle$.

Consider the case where there is another state $s \neq s_1$ such that $s \sim_S s_1$. Let A be an event containing s_1 but not s . By monotony of σ , $s_1 \succ_S A$. By contradiction, assume $s \sim_S s_1 \prec_S A$. Applying **NES'** with $A, B = s, C = s$, we get $s \prec_S A \cap s = \emptyset$, which is a contradiction. Then any event containing some states equivalent to s_1 but not all those states is equivalent to s_1 . Consider the event B containing all those states. The complement of B is the event containing all the states that are less plausible than s_1 . Then by **AD**, $B \succ_S s_1$. Assume there is k equivalence classes in D^{\prec_S} . Then 2^S has $2k + 1$ equivalence classes. Let $\sigma(D) = \langle 1_L, 1_L \rangle$.

Now consider the opposite case where there is only one state s_1 . The complement of s_1 is the event containing all the states that are less plausible than s_1 . Then 2^S has $2k$ equivalence classes. In both cases, by **AD**, σ is extended on D^{\succ_S} . \square

Remark that the previous theorem could have been stated with axiom **NES'** in place of **POS'**.

4 Axiomatics

For a Sugeno integral to be a binary possibilistic utility, the only condition is to ensure that the capacity is an autodual possibilistic capacity.

For any act $h \in \mathcal{F}$, we write $h^{\prec\mathcal{F}}$ for the set of acts $\{f \in \mathcal{F} : f \prec_{\mathcal{F}} h\}$ and $h^{\succ\mathcal{F}}$ for $\{f \in \mathcal{F} : f \succ_{\mathcal{F}} h\}$. They are respectively the set of acts that are less preferred than h and its complement.

The following axioms are required to state our theorem:

$$\begin{aligned} \textbf{AD } & \forall A, B \in 2^S, 1_X A 0_X \succ_{\mathcal{F}} 1_X B 0_X \Leftrightarrow 1_X \bar{B} 0_X \succ_{\mathcal{F}} 1_X \bar{A} 0_X \\ \textbf{OPT' } & \exists h \in \mathcal{F}, 1_X \succ_{\mathcal{F}} h \succ_S 0_X \text{ such that } \forall f, g \in h^{\succ\mathcal{F}}, \forall A \in 2^S, f \succ_{\mathcal{F}} f A g \Rightarrow g A f \succ_{\mathcal{F}} f \end{aligned}$$

Axiom **AD** is a reformulation of axiom **AD**^S in the context of acts. Axiom **OPT'** states that the set of acts can be divided into two parts, where the set of less preferred acts satisfies axiom **OPT**.

The following representation theorem for PU can be stated.

Theorem 6. If $\succ_{\mathcal{F}}$ is a preorder over \mathcal{F} , satisfies **Sav1**, **WS3**, **Sav5**, **RCD**, **RDD**, **OPT'** and **AD** then there is a finite qualitative scale (L, \geq_L) , a basic utility assignment $u : X \rightarrow L_2$ and a possibilistic distribution $\pi : 2^S \rightarrow L$ such that $f \succ_{\mathcal{F}} f' \Leftrightarrow P_U(f) \geq_{L_2} P_U(f')$.

Proof. Axioms **Sav1**, **WS3**, **Sav5** and **OPT'** implies axiom **POS'**. See Theorem 4 of [11]. Then the result follows from Proposition 1 and Theorems 2 and 5. \square

What is remarkable in this result is that although PU is an autodual criterion, the initial preference relation over acts does not have to be autodual. Indeed only the autoduality for the representation of uncertainty is required and no structure is assumed over the set of consequences X . This shows that PU could be applied even when the consequences are not completely symmetrical around a middle point.

With the following axiom of autoduality of the preference relation over acts, a simplified theorem can be formulated:

AD+ Preference relation $\succ_{\mathcal{F}}$ over \mathcal{F} is autodual

Theorem 7. If $\succ_{\mathcal{F}}$ is a preorder over \mathcal{F} , satisfies **Sav1**, **WS3**, **Sav5**, **RCD**, **OPT'** and **AD+** then there is a finite qualitative scale (L, \geq_L) , a basic utility assignment $u : X \rightarrow L_2$ and a possibilistic distribution $\pi : 2^S \rightarrow L$ such that $f \succ_{\mathcal{F}} f' \Leftrightarrow P_U(f) \geq_{L_2} P_U(f')$.

Proof. Axiom **RDD** follows from **RCD** and **AD+**. Axiom **AD+** implies **AD**. Then the previous theorem can be applied. \square

Compared to Theorem 6, this theorem gives a better picture of the properties of PU. Thanks to the autoduality of PU, in the two previous theorem, axiom **OPT'** could be replaced by **PES'**.

5 CONCLUSION

In this paper we gave an axiomatization à la Savage for the binary possibilistic utility, which is the qualitative counterpart of expected utility. This criterion enjoys many properties similar to its quantitative counterpart: autoduality of the underlying uncertainty measure, autoduality of the decision criterion and dynamic consistency. Moreover being the unification of optimistic and pessimistic utilities, PU

can model different decision attitudes (by shifting utilities of consequences) as can be expected with convex and concave utilities.

However as a qualitative model, it sometimes lacks decisiveness. In [27], a new qualitative model, refined binary possibilistic utility, which is an improvement over PU has been proposed. As a future work, a Savagean axiomatic study could be carried on with this refined decision model in order to unveil its properties.

REFERENCES

- [1] M. Allais, ‘Le comportement de l’homme rationnel devant le risque : critique des postulats de l’école américaine’, *Econometrica*, **21**, 503–546, (53).
- [2] C. Boutilier, ‘Toward a logic for qualitative decision theory’, in *KR*, volume 2, pp. 75–86, (1994).
- [3] R.I. Brafman and M. Tennenholtz, ‘On the axiomatization of qualitative decision criteria’, in *AAAI*, volume 14, pp. 76–81, (1997).
- [4] F.C. Chu and J.Y. Halpern, ‘Great expectations. part i: On the customizability of generalized expected utility’, in *IJCAI*, volume 18, pp. 291–296, (2003).
- [5] D. Dubois, ‘Belief structures, possibility theory and decomposable confidence measures on finite sets’, *Computers and Artificial Intelligence*, **5**, 401–416, (1986).
- [6] D. Dubois, L. Godo, H. Prade, and A. Zapico, ‘Making decision in a qualitative setting: from decision under uncertainty to case-based decision’, in *KR*, volume 6, pp. 594–607, (1998).
- [7] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*, Academic press, 1980.
- [8] D. Dubois and H. Prade, ‘An introduction to possibilistic and fuzzy logics’, in *Readings in Uncertain Reasoning*, 742–761, Morgan Kaufmann, (1990).
- [9] D. Dubois and H. Prade, ‘Possibility theory as a basis of qualitative decision theory’, in *IJCAI*, volume 14, pp. 1925–1930, (1995).
- [10] D. Dubois, H. Prade, and R. Sabbadin, ‘Qualitative decision theory with sugeno integrals’, in *UAI*, volume 14, pp. 121–128, (1998).
- [11] D. Dubois, H. Prade, and R. Sabbadin, ‘Decision-theoretic foundations of qualitative possibility theory’, *European Journal of Operational Research*, **128**, 459–478, (2001).
- [12] D. Ellsberg, ‘Risk, ambiguity, and the savage axioms’, *Quarterly Journal of Economics*, **75**, 643–669, (1961).
- [13] N. Friedman and J. Halpern, ‘Plausibility measures: A user’s guide’, in *UAI*, volume 11, pp. 175–184, (1995).
- [14] P.H. Giang and P.P. Shenoy, ‘A qualitative linear utility theory for spohn’s theory of epistemic beliefs’, in *UAI*, volume 16, pp. 220–229, (2000).
- [15] P.H. Giang and P.P. Shenoy, ‘A comparison of axiomatic approaches to qualitative decision making using possibility theory’, in *UAI*, volume 17, pp. 162–170, (2001).
- [16] P.H. Giang and P.P. Shenoy, ‘Two axiomatic approaches to decision making using possibility theory’, *European Journal of Operational Research*, **162**(2), 450–467, (2005).
- [17] I. Gilboa, ‘Expected utility with purely subjective non-additive probabilities’, *Journal of Mathematical Economics*, **16**, 65–88, (1987).
- [18] J.Y. Jaffray and P. Wakker, ‘Decision making with belief functions: Compatibility and incompatibility with the sure-thing principle’, *J. of Risk and Uncertainty*, **7**(3), 255–71, (1993).
- [19] J. Pearl, ‘From conditional oughts to qualitative decision theory’, in *UAI*, volume 9, pp. 12–22, (1993).
- [20] P. Perny, O. Spanjaard, and P. Weng, ‘Algebraic markov decision processes’, in *IJCAI*, volume 19, pp. 1372–1377, (2005).
- [21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 2nd edn., 2003.
- [22] L.J. Savage, *The foundations of statistics*, J. Wiley and sons, 1954.
- [23] D. Schmeidler, ‘Subjective probability and expected utility without additivity’, *Econometrica*, **57**, 571–587, (1989).
- [24] G. Shafer, *A mathematical theory of evidence*, Princeton university press, 1976.
- [25] W. Spohn, ‘A general non-probabilistic theory of inductive reasoning’, in *UAI*, volume 4, (1988).
- [26] J. von Neumann and O. Morgenstern, *Theory of games and economic behavior*, Princeton university press, 1944.
- [27] Paul Weng, ‘Qualitative decision making under possibilistic uncertainty: Toward more discriminating criteria’, in *UAI*, volume 21, (2005).

An Efficient Upper Approximation for Conditional Preference

Nic Wilson¹

Abstract. The fundamental operation of dominance testing, i.e., determining if one alternative is preferred to another, is in general very hard for methods of reasoning with qualitative conditional preferences such as CP-nets and conditional preference theories (CP-theories). It is therefore natural to consider approximations of preference, and upper approximations are of particular interest, since they can be used within a constraint optimisation algorithm to find some of the optimal solutions. Upper approximations for preference in CP-theories have previously been suggested, but they require consistency, as well as strong acyclicity conditions on the variables. We define an upper approximation of conditional preference for which dominance checking is efficient, and which can be applied very generally for CP-theories.

1 Introduction

A basic operation for a preference formalism is testing dominance, i.e., checking if one alternative is preferred to another. Unfortunately this has been shown to be very hard (PSPACE-complete) [7] in general for CP-nets [1, 2] and conditional preference theories (CP-theories) [13, 12], two formalisms for reasoning with qualitative and comparative conditional preferences; the cases where it is known to be feasible are of a very simple form [6, 2].

Dominance testing has particular importance for constrained optimisation; the algorithm for constrained optimisation given in [3] involves dominance testing if one requires more than one undominated solution of a set of constraints, and can involve a great many dominance checks; similar problems apply to the approach in [11]. This problem can be side-stepped by using dominance checking with respect to an upper approximation of preference (see e.g., [13]). If a solution is undominated with respect to the upper approximation it ensures that it will be undominated with respect to the preference relation; the algorithm of [3] amended in this way then generates *some* undominated solutions, but usually not all of them. Dominance testing with respect to such an upper approximation needs to be fast, since many such tests will typically be required. However, it is often not essential that the upper approximation be a close approximation, since we can usually afford to lose undominated solutions—in many situations there will be huge numbers of them, and so we would not be able to explicitly list them all even if we could find them.

Efficient upper approximations have been defined in [13, 12], but they require restrictive conditions on the CP-theory: consistency and strong acyclicity properties on the variables used in the conditional preference statements. However there are many natural situations when these conditions are not satisfied, even for simple examples,

see e.g., [5, 10, 11, 7]. For larger problems, it will be inconvenient and often impractical to have to restrict a user's preference statements so that the CP-theory is of such a special form, firstly, because this may very well mean they cannot express the preferences they wish to, and, secondly, because checking consistency of a CP-net or a CP-theory can be very hard [7].

The main contribution of this paper is to define an upper approximation of conditional preference, for which dominance testing is efficient, and which can be applied very widely for CP-theories; i.e., without the strong acyclicity properties on variables, and without assuming consistency. The approximation is also a closer one than previous upper approximations.

Section 2 describes conditional preference theories, as defined in [13, 12] which is an approach for reasoning with conditional preferences generalising CP-nets and TCP-nets [4]; a new semantics in terms of total pre-orders is also given. Section 3 describes *pre-ordered search trees* with their associated total pre-orders. Our new upper approximation is defined as the intersection of all those total pre-orders satisfying the CP-theory which arise from a pre-ordered search tree. Section 4 presents a number of technical results giving equivalent forms of these definitions; this leads to a simple and efficient algorithm for testing dominance with respect to the upper approximation. Section 5 discusses comparisons with other upper approximations and constrained optimisation.

2 Conditional Preference Theories

Let V be a set of n variables. For each $X \in V$ let \underline{X} be the set of possible values of X ; we assume \underline{X} has at least two elements. For subset of variables $U \subseteq V$ let $\underline{U} = \prod_{X \in U} \underline{X}$ be the set of possible assignments to set of variables U . The assignment to the empty set of variables is written \top . An *outcome* is an element of \underline{V} , i.e., an assignment to all the variables. For partial tuples $a \in \underline{A}$ and $u \in \underline{U}$, we may write $a \models u$, or say a extends u , if $A \supseteq U$ and $a(U) = u$, i.e., a projected to U gives u . More generally, we say that a is compatible with u if there exists outcome $\alpha \in \underline{V}$ extending both a and u , i.e., such that $\alpha(A) = a$ and $\alpha(U) = u$.

The language \mathcal{L}_V (abbreviated to \mathcal{L}) consists of statements of the form $u : x > x' [W]$ where u is an assignment to set of variables $U \subseteq V$ (i.e., $u \in \underline{U}$), x, x' are different values of variable X , and $\{X\}, U$ and W are pairwise disjoint. Let $T = V - (\{X\} \cup U \cup W)$. Such a conditional preference statement φ is intended to represent that given u and any assignment to T , x is preferred to x' irrespective of the values of W . This informal meaning is captured by the set φ^* of pairs of outcomes $\{(tuxw, tux'w') : t \in \underline{T}, w, w' \in \underline{W}\}$, (with $(tuxw, tux'w')$ meaning that $tuxw$ is preferred to $tux'w'$) since u is satisfied in both outcomes $tuxw$ and $tux'w'$, and variable X has the value x in the first, and x' in the second, and they differ at most on

¹ Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Cork, Ireland, n.wilson@4c.ucc.ie

$\{X\} \cup W$. We also say that $tux'w'$ is a *worsening swap from tuxw* (with respect to φ). So, pairs (α, β) in φ^* are intended to represent a preference for α over β , and statement φ is intended as a compact representation of the preference information φ^* .

Subsets Γ of the language \mathcal{L} are called *conditional preference theories (CP-theories)* [13]. For CP-theory Γ , define Γ^* to be $\bigcup_{\varphi \in \Gamma} \varphi^*$, which represents a set of preferences. We suppose here that preferences should be transitive, so it is then natural to define the associated order \succ_Γ , induced on \underline{V} by Γ , to be the transitive closure of Γ^* . So α is preferred to β , i.e., $\alpha \succ_\Gamma \beta$, if and only if there is a sequence of worsening swaps from α to β (each with respect to some element of Γ ; see [13]). Relation \succeq_Γ is defined by $\alpha \succeq_\Gamma \beta$ if and only if either $\alpha = \beta$ or $\alpha \succ_\Gamma \beta$. If φ is the statement $u : x > x' [W]$ and $u \in \underline{U}$ we may write $u_\varphi = u$, $U_\varphi = U$, $x_\varphi = x$, $x'_\varphi = x'$, $W_\varphi = W$. If W is empty we may omit $[W]$, writing just $u : x > x'$.

We say that outcome β *dominates* α (with respect to \succ_Γ) if $\beta \succ_\Gamma \alpha$. Outcome α is said to be *undominated* (w.r.t. \succ_Γ) if there exists no outcome that dominates it. (Although in this paper we focus on this strong sense of *undominated*, of interest also are outcomes α which are dominated only by outcomes which α dominates.) Finding an undominated outcome of a CP-theory amounts to finding a solution of a CSP [12] (so checking if there exists an undominated outcome is an NP-complete problem). Solution α to a set of constraints C (involving variables V) is said to be undominated if it is not dominated by any other solution of C . Finding undominated solutions seems to be a harder problem.

This is a relatively expressive language of conditional preferences: CP-nets [1, 2] can be represented by a set of statements of the form $u : x > x' [W]$ with $W = \emptyset$ [13], and TCP-nets [4] can be represented in terms of such statements with $W = \emptyset$ or $|W| = 1$ [12]. Lexicographic orders can also be represented [13]. Other related languages for conditional preferences include those in [8, 9].

Example. Let V be the set of variables $\{X, Y, Z\}$ with domains as follows: $\underline{X} = \{\bar{x}, \bar{\bar{x}}\}$, $\underline{Y} = \{\bar{y}, \bar{\bar{y}}\}$ and $\underline{Z} = \{\bar{z}, \bar{\bar{z}}\}$. Let $\varphi_1 = \top : x > \bar{x}$, let $\varphi_2 = x : y > \bar{y}$, let $\varphi_3 = y : z > \bar{z}$, and let $\varphi_4 = \bar{x} : \bar{z} > z$. Let CP-theory Γ be $\{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$. This gives rise to the following preferences, where e.g., $xyz \succ^3 \bar{x}\bar{y}\bar{z}$ means that xyz is a worsening swap from $\bar{x}\bar{y}\bar{z}$ with respect to φ_3 (in other words, $(xyz, \bar{x}\bar{y}\bar{z}) \in \varphi_3^*$): $xyz \succ^3 xyz \succ^2 \bar{x}\bar{y}\bar{z} \succ^1 \bar{x}\bar{y}\bar{z} \succ^4 \bar{x}yz$. We also have $xyz \succ^2 \bar{x}yz \succ^1 \bar{x}\bar{y}z$; and $xyz \succ^1 \bar{x}yz$, and $xyz \succ^1 \bar{x}\bar{y}\bar{z}$. In addition there is the cycle $\bar{x}\bar{y}\bar{z} \succ^4 \bar{x}yz \succ^3 \bar{x}\bar{y}\bar{z}$. The relation \succ_Γ is the transitive closure of these orderings. The cycle shows that Γ is inconsistent (see below). But, despite this “localised” inconsistency (which only involves two outcomes), useful things can be said. In particular, there is a single undominated outcome: xyz . Furthermore, if we add the constraint $(Y = \bar{y}) \vee (Z = \bar{z})$ then there are two undominated solutions: $\bar{x}\bar{y}\bar{z}$ and $\bar{x}\bar{y}z$. In this case the constraint also “removes the inconsistency” in the following sense: \succ_Γ restricted to solutions is acyclic.

Some general properties of relations

A relation \succcurlyeq on a set A is formally defined to be a set of pairs, i.e., a subset of $A \times A$. We will often write $a \succcurlyeq b$ to mean $(a, b) \in \succcurlyeq$. Relation \succ on set A is irreflexive if and only if for all $a \in A$, it is not the case that $a \succ a$. A pre-order \succcurlyeq on A is a reflexive and transitive relation, i.e., such that $a \succcurlyeq a$ for all $a \in A$, and such that $a \succcurlyeq b$ and $b \succcurlyeq c$ implies $a \succcurlyeq c$. Elements a and b are said to be \succcurlyeq -equivalent if $a \succcurlyeq b$ and $b \succcurlyeq a$. We may write \succ for the *strict part of* \succcurlyeq , i.e., the relation given by $a \succ b$ if and only if $a \succcurlyeq b$ and $b \not\succcurlyeq a$. (Note,

however, that \succ_Γ is not necessarily the strict part of \succeq_Γ .) A relation \succcurlyeq is complete if for all $a \neq b$, either $a \succcurlyeq b$ or $b \succcurlyeq a$. Relation \succcurlyeq is anti-symmetric if $a \succcurlyeq b$ and $b \succcurlyeq a$ implies $a = b$ (i.e., iff \succcurlyeq -equivalence is no more than equality). A partial order is an anti-symmetric pre-order. A total pre-order is a complete pre-order. If \succcurlyeq is a total pre-order then $a \succ b$ if and only if $b \not\succcurlyeq a$. A total order is a complete partial order. We say that a relation is acyclic if its transitive closure is anti-symmetric, i.e., there are no cycles $a \succcurlyeq a' \cdots \succcurlyeq a$ for different a, a', \dots . A relation \succcurlyeq' extends (or, alternatively, contains) relation \succcurlyeq if $\succcurlyeq' \supseteq \succcurlyeq$, i.e., if $a \succcurlyeq' b$ holds whenever $a \succcurlyeq b$ holds.

Semantics

Sequences of worsening swaps can be considered as a proof theory for this system of conditional preference. In [13] a semantics is given in terms of total orders (based on the semantics for CP-nets [2]); in the case of an inconsistent CP-theory, the semantic entailment relation becomes degenerate. To deal with this problem, [5] defines an extended semantics for CP-nets in terms of total pre-orders. We show how this semantics can be generalised to CP-theories, extending also the semantics in [13].

Let \succcurlyeq be a total pre-order on \underline{V} . We say that \succcurlyeq satisfies conditional preference statement φ if $\alpha \succcurlyeq \beta$ holds whenever β is a worsening swap from α w.r.t. φ ; this is if and only if \succcurlyeq extends the relation φ^* . We say that \succcurlyeq satisfies a CP-theory Γ if it satisfies each element φ of Γ , i.e., if \succcurlyeq extends the relation Γ^* . Because \succcurlyeq is transitive, this holds if and only if \succcurlyeq extends \succ_Γ .

For different outcomes α and β we say that $\Gamma \models' (\alpha, \beta)$ if $\alpha \succcurlyeq \beta$ holds for all total pre-orders \succcurlyeq satisfying Γ . We also say, as in [13], that $\Gamma \models (\alpha, \beta)$ if $\alpha \succcurlyeq \beta$ holds for all total orders \succcurlyeq satisfying Γ . We say that Γ is *consistent* if there exists some *total order* satisfying Γ .

The following theorem² shows that swapping sequences are complete with respect to the semantics based on total pre-orders. Also, in the case of consistent CP-theories, the two semantic consequence relations are equivalent.

Theorem 1 Let $\Gamma \subseteq \mathcal{L}_V$ be a CP-theory and let $\alpha, \beta \in \underline{V}$ be different outcomes. Then (i) the relation \succ_Γ is the intersection of all total pre-orders satisfying Γ ; (ii) $\Gamma \models' (\alpha, \beta)$ if and only if $\alpha \succ_\Gamma \beta$; (iii) Γ is consistent if and only if \succ_Γ is irreflexive if and only if \succeq_Γ is a partial order; (iv) if Γ is consistent then $\Gamma \models (\alpha, \beta)$ if and only if $\Gamma \models' (\alpha, \beta)$.

The semantics suggests a general approach to finding an upper approximation of \succ_Γ : we consider a subset \mathcal{M} of the set of all total pre-orders (which might be thought of as a set of “preferred” models) and define that α is preferred to β (with respect to this approximation) if $\alpha \succcurlyeq \beta$ for all \succcurlyeq in \mathcal{M} which satisfy Γ . (It is an “upper approximation” in the sense that the approximation contains the relation \succ_Γ .) We use this kind of approach in the next section.

3 Pre-ordered Search Trees

A pre-ordered search tree is a rooted directed tree (which we imagine being drawn with the root at the top, and children below parents). Associated with each node r in the tree is a variable Y_r , which is instantiated with a different value in each of the node’s children (if it

² The proof of this result, and proofs of other results in the paper are included in a longer version of the paper available at the 4C website: <http://www.4c.ucc.ie/>.

has any), and also a pre-ordering \geq_r of the values of Y_r . A directed edge in the tree therefore corresponds to an instantiation of one of the variables to a particular value. Paths in the tree from the root down to a leaf node correspond to sequential instantiations of different variables. We also associate with each node r a set of variables A_r which is the set of all variables $Y_{r'}$ associated to nodes r' above r in the tree (i.e., on the path from the root to r), and an assignment a_r to A_r corresponding to the assignments made to these variables in the edges between the root and r . The root node r^* has $A_{r^*} = \emptyset$ and $a_{r^*} = \top$, the assignment to the empty set of variables. Hence r' is a child of r if and only if $A_{r'} = A_r \cup \{Y_r\}$ (where $A_r \not\ni Y_r$) and $a_{r'}$ extends a_r (with an assignment to Y_r).

More formally, define a node r to be a tuple $\langle A_r, a_r, Y_r, \geq_r \rangle$, where $A_r \subseteq V$ is a set of variables, $a_r \in A_r$ is an assignment to those variables, $Y_r \in V - A_r$ is another variable, and \geq_r is a total pre-order on the set \underline{Y}_r of values of Y_r . We make two restrictions on the choice of this total pre-order: firstly, it is assumed not to be the trivial complete relation on \underline{Y} ; so there exists some $y, y' \in \underline{Y}$ with $y \not\geq_r y'$. We also assume that total pre-order \geq_r satisfies the following condition (which we require so that the associated ordering on outcomes is transitive): if there exists a child of node r associated with instantiation $Y_r = y$, then y is not \geq_r -equivalent to any other value of Y , so that $y \geq_r y' \geq_r y$ only if $y' = y$. In particular, \geq_r totally orders the values (of Y_r) associated with the children of r .

For outcome α , define the *path to α* to be the path from the root which includes all nodes r such that α extends a_r . To generate this, for each node r , starting from the root, we choose the child associated with the instantiation $Y_r = \alpha(Y_r)$ (there is at most one such child); the path finishes when there exists no such child.

Node r is said to *decide outcomes α and β* if it is the deepest node (i.e., furthest from the root) which is both on the path to α and on the path to β . Hence α and β both extend the tuple a_r (but they may differ on variable Y_r). We compare α and β by using \geq_r , where r is the unique node which decides α and β .

Each pre-ordered search tree σ has an associated ordering relation \succ_σ on outcomes which is defined as follows. Let $\alpha, \beta \in \underline{V}$ be outcomes. We define $\alpha \succ_\sigma \beta$ to hold if and only if $\alpha(Y_r) \geq_r \beta(Y_r)$, where r is the node which decides α and β . We therefore then have that α and β are \succ_σ -equivalent if and only if $\alpha(Y_r)$ and $\beta(Y_r)$ are \geq_r -equivalent; also: $\alpha \succ_\sigma \beta$ holds if and only if $\alpha(Y_r) >_r \beta(Y_r)$. This ordering is similar to a lexicographic ordering in that two outcomes are compared on the first variable on which they differ.

The definition implies immediately that \succ_σ is reflexive and complete; it can be shown easily that it is also transitive. Hence it is a total pre-order. We say that pre-ordered search tree σ *satisfies* conditional preference theory Γ iff \succ_σ satisfies Γ .

Definition of Upper Approximation. For a given CP-theory Γ we define the relation \sqsupseteq_Γ (abbreviated to \sqsupseteq) as follows: $\alpha \sqsupseteq \beta$ holds if and only if $\alpha \succ_\sigma \beta$ holds for all σ satisfying Γ (i.e., all σ such that \succ_σ satisfies Γ). Relation \sqsupseteq is then the intersection of all pre-ordered search tree orders which satisfy Γ . The intersection of a set of reflexive and transitive relations containing \succ_Γ is clearly reflexive and transitive, and contains \succ_Γ :

Proposition 1 For any CP-theory Γ , the relation \sqsupseteq_Γ is a pre-order which is an upper approximation of \succ_Γ , i.e., if $\alpha \succ_\Gamma \beta$ then $\alpha \sqsupseteq_\Gamma \beta$.

This implies, in particular, that if outcomes α and β are incompatible with respect to \sqsupseteq_Γ then they are incomparable with respect to \succ_Γ .

Example continued. Consider the pre-ordered search tree σ_1 with just one node, the root node $r = \langle \emptyset, \top, Y, y > \bar{y} \rangle$. Let α and β be any outcomes with $\alpha(Y) = y$ and $\beta(Y) = \bar{y}$. We then have $\alpha \succ_{\sigma_1} \beta$ since the node r decides α and β , and $\alpha(Y) >_r \beta(Y)$. In particular, this implies that σ_1 satisfies φ_2 . If outcomes γ and δ agree on variable Y , then γ and δ are \succ_{σ_1} -equivalent because $\gamma(Y) = \delta(Y)$. Hence σ_1 satisfies φ_1, φ_3 and φ_4 and so satisfies Γ . This implies that $\beta \not\sqsupseteq_\Gamma \alpha$, so, in particular, $xy\bar{z} \not\sqsupseteq xy\bar{z}$.

Now consider the pre-ordered search tree σ_2 which has root node $\langle \emptyset, \top, X, x > \bar{x} \rangle$ with a single child node $\langle \{X\}, x, Z, z > \bar{z} \rangle$. This also satisfies Γ . For example, to check that σ_2 satisfies φ_4 we can reason as follows: let α and β be any outcomes such that $(\alpha, \beta) \in \varphi_4^*$, so that $\alpha(X) = \beta(X) = \bar{x}$, $\alpha(Y) = \beta(Y) = \bar{z}$, $\alpha(Z) = \bar{z}$ and $\beta(Z) = z$. Then the root node $r' = \langle \emptyset, \top, X, x > \bar{x} \rangle$ decides α and β because its single child is associated with $X = x$, which is incompatible with α and β . Now, $Y_{r'} = X$ and α and β agree on X so $\alpha(X)$ and $\beta(X)$ are $\geq_{r'}$ -equivalent; in particular, $\alpha(X) \geq_{r'} \beta(X)$. Hence $\alpha \succ_{\sigma_2} \beta$. Pre-ordered search tree σ_2 strictly prefers $x\bar{y}z$ to $xy\bar{z}$, which shows that $xy\bar{z} \not\sqsupseteq x\bar{y}z$. We've shown that both \succ_Γ -undominated solutions are also \sqsupseteq -undominated. In fact, in this case, \sqsupseteq_Γ is actually equal to \succ_Γ .

4 Computation of Upper Bound on Preference

Outcome α is \sqsupseteq -preferred to β if and only if α is preferred to β in all pre-ordered search trees satisfying Γ . At first sight this definition looks computationally very unpromising for two reasons: (i) direct testing of whether a pre-ordered search tree satisfies Γ is not feasible, as Γ^* will typically contain exponentially many pairs; (ii) there will very often be a huge number of pre-ordered search trees satisfying Γ .

In this section we find ways of getting round these two problems. We first (Section 4.1) find simpler equivalent conditions for a pre-ordered search tree σ to satisfy Γ ; then, in 4.2, we use the results of 4.1 to recast the problem of testing dominance with respect to the upper approximation, allowing a simple and efficient algorithm.

4.1 Equivalent conditions for σ to satisfy Γ

Consider a pre-ordered search tree σ , and let α be any outcome. Associated with the path to α is the sequence of variables Y_1, \dots, Y_k which are instantiated along that path (i.e., associated with the nodes on the path), starting with the root node. If W is a set of variables and X is a variable not in W , we say that “*on the path to α , X appears before any of W* ” if the following condition holds: $Y_j \in W$ implies that $Y_i = X$ for some $i < j$, i.e., if some element of W occurs on the path then X occurs earlier on the path.

Proposition 2 The following pair of conditions are necessary and sufficient for a pre-ordered search tree σ to satisfy the CP-theory Γ :

- (1) For any $\varphi \in \Gamma$ and outcome α extending u_φ : on the path to α , X_φ appears before W_φ ;
- (2) for any node r and any $\varphi \in \Gamma$ with $X_\varphi = Y_r$ we have $x_\varphi \geq_r x'_\varphi$ if u_φ is compatible with a_r .

Relation \sqsupseteq_a^X . Because \geq_r is transitive, condition (2) can be written equivalently as: for all nodes r in σ , $\geq_r \supseteq \sqsupseteq_a^X$, where \sqsupseteq_a^X is defined to be the transitive closure of the set of pairs (x, x') of values of X over all statements $u : x > x' [W]$ in Γ such that u is compatible with a . Note that relation \sqsupseteq_a^X is monotonic decreasing

with respect to a : if b extends a (to variables not including X) then \square_a^X contains \square_b^X ; this is because if u_φ is compatible with b then u_φ is compatible with a .

Let $A \subseteq V$ be a set of variables, let $a \in \underline{A}$ be an assignment to A and let $Y \in V - A$ be some variable not in A . We say that Y is *a-choosable* if $X_\varphi \in A$ for all $\varphi \in \Gamma$ satisfying (a) u_φ compatible with a and (b) $W_\varphi \ni Y$. Note that if $W_\varphi = \emptyset$ for all $\varphi \in \Gamma$, as in CP-nets, then, for all a , every variable is *a-choosable*. If we are attempting to construct a node r of a pre-ordered search tree satisfying Γ , where a_r is the associated assignment, then we need to pick a variable Y_r which is a_r -choosable, because of Proposition 2(1). This condition has the following monotonicity property: suppose $A \subseteq B \subseteq V$, and $Y \notin B$; suppose also that b is an assignment to B extending assignment a to variables A . Then Y is b -choosable if Y is a -choosable.

Define pre-ordered search tree node r to satisfy Γ if Y_r is a_r -choosable and \geq_r satisfies condition (2) above, i.e., $\geq_r \supseteq \square_{a_r}^{Y_r}$. It can be seen that Y_r is a_r -choosable for each node r in pre-ordered search tree σ if and only if condition (1) of Proposition 2 is satisfied. This leads to the following result.

Proposition 3 *A pre-ordered search tree σ satisfies Γ if and only if each node of σ satisfies Γ .*

4.2 Computation of upper bound on preference relation

In this section we consider a fixed conditional preference theory Γ . Suppose we are given outcomes α and β , and we wish to determine if $\beta \trianglerighteq \alpha$ or not. By definition, $\beta \not\trianglerighteq \alpha$ if and only if there exists a pre-ordered search tree σ satisfying Γ with $\beta \not\succ_\sigma \alpha$, i.e., with $\alpha \succ_\sigma \beta$. Therefore, an approach to showing $\beta \not\trianglerighteq \alpha$ is to construct a pre-ordered search tree σ with $\alpha \succ_\sigma \beta$. The key is to construct the path from the root which will form the intersection of the path to α and the path to β ; for each node r on this path we need to choose a variable Y_r with certain properties. If α and β differ on Y_r it needs to be possible to choose the local relation \geq_r so that $\alpha(Y_r) >_r \beta(Y_r)$. If α and β agree on Y_r then we need to ensure that the local relation is such that this node can have a child. With this in mind we make the following definitions.

Suppose α and β are both outcomes which extend partial tuple $a \in \underline{A}$. Define variable Y to be *pickable given a with respect to* (α, β) if $Y \notin A$ and (i) Y is *a-choosable*; (ii-a) if $\alpha(Y) = \beta(Y)$ then $\alpha(Y)$ is not \square_a^Y -equivalent to any other value in \underline{Y} ; (ii-b) if $\alpha(Y) \neq \beta(Y)$ then $\beta(Y) \not\succ_a^Y \alpha(Y)$. If Y is pickable given a with respect to (α, β) , and $\alpha(Y) \neq \beta(Y)$ then we say that Y is *decisive given a (with respect to (α, β))*.

The following lemma describes a key monotonicity property. It follows immediately from the previously observed monotonicity properties of being *a-choosable*, and of \square_a^Y .

Lemma 1 *Let α and β be two outcomes which both extend tuples a and b , where $a \in \underline{A}$ and $b \in \underline{B}$ and $A \subseteq B \subseteq V$ (so b extends a). Let Y be a variable not in B . If Y is pickable given a with respect to (α, β) then Y is pickable given b with respect to (α, β) .*

A *decisive sequence* (w.r.t. (α, β)) is defined to be a sequence Y_1, \dots, Y_k of variables satisfying the following three conditions:

- for $j = 1, \dots, k-1$, $\alpha(Y_j) = \beta(Y_j)$,
- $\alpha(Y_k) \neq \beta(Y_k)$
- for $j = 1, \dots, k$, Y_j is pickable given a_j (with respect to (α, β)), where a_j is α restricted to $\{Y_1, \dots, Y_{j-1}\}$; in particular, Y_k is decisive given a_k .

Proposition 4 *There exists a decisive sequence w.r.t. (α, β) if and only if there exists a pre-ordered search tree σ satisfying Γ with $\alpha \succ_\sigma \beta$.*

Since $\beta \triangleright \alpha$ holds if and only if there exists no pre-ordered search tree σ satisfying Γ with $\alpha \succ_\sigma \beta$, Proposition 4 implies the following result.

Proposition 5 *For outcomes α and β , $\beta \triangleright \alpha$ holds if and only if there exists no decisive sequence with respect to (α, β) .*

Therefore to determine if $\beta \triangleright \alpha$ or not, we just need to check if there exists a decisive sequence Y_1, \dots, Y_k . The monotonicity lemma implies that we do not have to be careful about the variable ordering: a variable which is pickable at one point in the sequence is still pickable at a later point; this means that we can choose, for each j , Y_j to be any pickable variable, knowing that we will not have to backtrack, as any previously available choices remain available later.

The following algorithm takes as input outcomes α and β and determines if $\beta \triangleright \alpha$ or not.

```

procedure Is  $\beta \triangleright \alpha$ ?
if  $\alpha = \beta$  then return true and stop;
for  $j := 1, \dots, n$ 
  let  $a_j$  be  $\alpha$  restricted to  $\{Y_1, \dots, Y_{j-1}\}$ ;
  if there exists a variable which is decisive given  $a_j$  w.r.t.  $(\alpha, \beta)$ 
    then return false and stop;
  if there exists a variable which is pickable given  $a_j$  w.r.t.  $(\alpha, \beta)$ 
    then let  $Y_j$  be any such variable;
    else return true and stop.

```

The correctness of the algorithm follows easily from Proposition 5 and the monotonicity lemma.

Theorem 2 *Let Γ be a CP-theory, and let α and β be outcomes. The above procedure is correct, i.e., it returns **true** if $\beta \triangleright_\Gamma \alpha$, and it returns **false** if $\beta \not\triangleright_\Gamma \alpha$.*

Example continued. Now let $\alpha = \bar{x}\bar{y}z$ and $\beta = \bar{x}\bar{y}\bar{z}$. Since for all $\varphi \in \Gamma$, $W_\varphi = \emptyset$, each variable is *a-choosable* for any a . The relation \square_\top^Z contains pair (z, \bar{z}) because of $\varphi_3 = y : z > \bar{z}$ (anything is compatible with $a = \top$). It also contains pair (\bar{z}, z) because of $\varphi_4 = \bar{x} : \bar{z} > z$ and so $\beta(Z) \sqsupseteq_\top^Z \alpha(Z)$ which implies that Z is not pickable given \top with respect to (α, β) since $\alpha(Z) \neq \beta(Z)$. On the other hand, X and Y are both pickable given \top (but not decisive). Suppose we select $Y_1 = Y$, and so $a_2 = \bar{y}$. Variable Z is still not pickable, but X is still pickable given a_2 (by Lemma 1) so we get $Y_2 = X$ and $a_3 = \bar{x}\bar{y}$. Relation $\square_{a_3}^Z$ is empty so Z is now pickable and hence decisive (giving decisive sequence Y, X, Z) proving that $\beta \not\triangleright \alpha$. A shorter decisive sequence is X, Z which corresponds to pre-ordered search tree σ_2 which strictly prefers α to β .

Complexity of approximate dominance checking We assume that the size $|\underline{X}|$ of the domain of each variable X is bounded by a constant; we will consider the complexity in terms of n , the number of variables, and of $m = |\Gamma|$, where we assume that m is at least $O(n)$. This algorithm can be implemented to have complexity at worst $O(mn^2)$, or, more precisely, $O(mn(w+1))$ where w is the average of $|W_\varphi|$ over $\varphi \in \Gamma$. Clearly $w < n$. For some special classes such as CP-theories representing CP-nets or TCP-nets, w is bounded by a constant, and so the complexity is then $O(mn)$.

5 Comparison and Discussion

An upper approximation $\succ_{p(\Gamma)}$ for \succ_Γ was defined in [13], which was refined to upper approximation \gg_Γ in [12]. It follows easily from their construction that $\gg_\Gamma \subseteq \succ_{p(\Gamma)}$ when the latter is defined.

Both these require consistency, and strong acyclicity properties on the ordering of variables used in statements in Γ . In particular, in both cases, it must be possible to label the set of variables V as $\{X_1, \dots, X_n\}$ in such a way that for any $\varphi \in \Gamma$, if $X_i \in U_\varphi$, and $X_j \in \{X_\varphi\} \cup W_\varphi$ then $i < j$. It can be proved using Proposition 5 that \sqsupseteq is never a worse upper approximation than \gg_Γ or $\succ_{p(\Gamma)}$.

Proposition 6 *Let α and β be two different outcomes such that $\alpha \sqsupseteq \beta$. Then $\alpha \gg_\Gamma \beta$ if Γ is such that \gg_Γ is defined; hence also $\alpha \succ_{p(\Gamma)} \beta$ if $\succ_{p(\Gamma)}$ is defined.*

Example continued. The upper approximations of [13, 12] are not applicable because Γ is not consistent. If we restore consistency by removing φ_4 from Γ then they are applicable but give a poorer upper approximation; in particular they both have $xy\bar{z}$ preferred to $x\bar{y}z$ (essentially because they make Y a more important variable than Z , as Y is a parent of Z), so that they miss undominated solution $x\bar{y}z$.

Application to constrained optimisation

In the constrained optimisation algorithm in [3], and the amendments in [13, 12], a search tree is used to find solutions, where the search tree is chosen so that its associated total ordering on outcomes extends \succ_Γ . Methods for finding such search trees have been developed in [12]. We can make use of this search tree as follows, amending the constrained optimisation approach of [12] in the obvious way: when we find a new solution α we check if it is \sqsupseteq -undominated with respect to each of the current known set K of \sqsupseteq -undominated solutions. If so, then α is an \sqsupseteq -undominated solution, since it cannot be \sqsupseteq -dominated by any solution found later. We add α to K , and continue the search. This is an anytime algorithm, but if we continue until the end of the search, K will be the complete set of \sqsupseteq -undominated solutions, which is a subset of the set of \succ_Γ -undominated solutions, since $\sqsupseteq \supseteq \succ_\Gamma$. For the inconsistent case, by definition, no such search tree can exist. Instead we could use a pre-ordered search tree satisfying Γ , and continue generating solutions with this for as long as it totally orders solutions. This may well be successful for cases where the inconsistency is relatively localised and among less preferred outcomes, such as in the example.

Crudeness of the approximation

As illustrated by the example, \sqsupseteq_Γ can be a close approximation of \succ_Γ . However, computing dominance with respect to \succ_Γ appears in general to be extremely hard [7] whereas our approximation \sqsupseteq_Γ is of low order polynomial complexity. One would therefore not expect \sqsupseteq_Γ always to be a close approximation. To illustrate this, consider outcomes α and β which differ on all variables (or, more generally on all variables not in $W_\Gamma = \bigcup_{\varphi \in \Gamma} W_\varphi$). Then any variable which is pickable is decisive, so, by Proposition 5, $\beta \sqsupseteq_\Gamma \alpha$ if and only if there exists no variable which is pickable given \top with respect to (α, β) . A variable is \top -choosable if and only if it is not in W_Γ , so $\beta \sqsupseteq_\Gamma \alpha$ if and only if for all $Y \in V - W_\Gamma$, $\beta(Y) \sqsupseteq_\top \alpha(Y)$. The relation \sqsupseteq_\top does not depend at all on u_φ , for $\varphi \in \Gamma$, so, for such α and β , whether $\beta \sqsupseteq_\Gamma \alpha$ holds or not does not depend at all on u_φ , for $\varphi \in \Gamma$. In particular, if $\beta \succ_{\Gamma_*} \alpha$ holds, where Γ_* is Γ in which each

u_φ is changed to \top , then, by Proposition 1, $\beta \sqsupseteq_\Gamma \alpha$ holds, since for such α and β , $\beta \sqsupseteq_\Gamma \alpha$ if and only if $\beta \sqsupseteq_{\Gamma_*} \alpha$. Since \succ_{Γ_*} can easily be a very crude upper approximation of \succ_Γ , this suggests that \sqsupseteq_Γ may often not be a close approximation for such pairs of outcomes, i.e., we may easily have $\beta \sqsupseteq_\Gamma \alpha$ without $\beta \succ_\Gamma \alpha$.

However, this does not necessarily matter for constrained optimisation. There will often be a very large number of optimal solutions, and we may well only wish to report a small fraction of them; it is not necessarily important that the upper approximation is a close approximation, just that \sqsupseteq is a sufficiently sparse (i.e., weak) relation, so that there are still liable to be a good number of solutions which are \sqsupseteq -undominated.

Summary

In this paper we have constructed a new upper approximation of conditional preference in CP-theories, which is very much more widely applicable than previous approaches, as well as being a better approximation. Furthermore, an efficient algorithm for dominance testing with respect to approximate preference has been derived.

ACKNOWLEDGEMENTS

This material is based upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075. I'm grateful to the referees for their helpful comments.

REFERENCES

- [1] C. Boutilier, R. Brafman, H. Hoos, and D. Poole, 'Reasoning with conditional *ceteris paribus* preference statements', in *Proceedings of UAI-99*, pp. 71–80, (1999).
- [2] C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'CP-nets: A tool for reasoning with conditional *ceteris paribus* preference statements', *Journal of Artificial Intelligence Research*, **21**, 135–191, (2004).
- [3] C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'Preference-based constrained optimization with CP-nets', *Computational Intelligence*, **20**(2), 137–157, (2004).
- [4] R. Brafman and C. Domshlak, 'Introducing variable importance trade-offs into CP-nets', in *Proceedings of UAI-02*, pp. 69–76, (2002).
- [5] R. I. Brafman and Y. Dimopoulos, 'Extended semantics and optimization algorithms for CP-networks', *Computational Intelligence*, **20**(2), 218–245, (2004).
- [6] C. Domshlak and R. I. Brafman, 'CP-nets—reasoning and consistency testing', in *Proc. KR02*, pp. 121–132, (2002).
- [7] J. Goldsmith, J. Lang, M. Truszczyński, and N. Wilson, 'The computational complexity of dominance and consistency in CP-nets', in *Proceedings of IJCAI-05*, pp. 144–149, (2005).
- [8] J. Lang, 'Logical preference representation and combinatorial vote', *Ann. Mathematics and Artificial Intelligence*, **42**(1), 37–71, (2004).
- [9] M. McGeachie and J. Doyle, 'Utility functions for *ceteris paribus* preferences', *Computational Intelligence*, **20**(2), 158–217, (2004).
- [10] S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh, 'Constrained CP-nets', in *Proceedings of CSCLP'04*, (2004).
- [11] S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh, 'Constraint-based preferential optimization', in *Proceedings of AAAI-05*, (2005).
- [12] N. Wilson, 'Consistency and constrained optimisation for conditional preferences', in *Proceedings of ECAI-04*, pp. 888–892, (2004).
- [13] N. Wilson, 'Extending CP-nets with stronger conditional preference statements', in *Proceedings of AAAI-04*, pp. 735–741, (2004).

A Solver for QBFs in Nonprenex Form¹

Uwe Egly² and Martina Seidl³ and Stefan Woltran²

Abstract. Various problems in artificial intelligence (AI) can be solved by translating them into a quantified boolean formula (QBF) and evaluating the resulting encoding. In this approach, a QBF solver is used as a black box in a rapid implementation of a more general reasoning system. Most of the current solvers for QBFs require formulas in prenex conjunctive normal form as input, which makes a further translation necessary, since the encodings are usually not in a specific normal form. This additional step increases the number of variables in the formula or disrupts the formula's structure. Moreover, the most important part of this transformation, prenexing, is not deterministic. In this paper, we focus on an alternative way to process QBFs without these drawbacks and describe a solver, qpro, which is able to handle arbitrary formulas. To this end, we extend algorithms for QBFs to the non-normal form case and compare qpro with the leading normal-form provers on problems from the area of AI.

1 Introduction

Formal frameworks are often suitable for the representation of application problems (like planning, scheduling, etc.) which can then be solved by automated reasoning tools. Many important problems in artificial intelligence (AI) (like problems in knowledge representation) can be encoded efficiently using quantified boolean formulas (QBFs), which are an extension of classical propositional formulas, permitting existential and universal quantifications over propositional atoms. QBFs have been proven to be a powerful framework for the rapid implementation of reasoning tasks from these areas (see, e.g., [6, 13, 24]), mainly because there has been made a significant progress in the development of QBF solvers in the last few years [19]. Almost all of these solvers expect the input formula to be in a certain *prenex conjunctive normal form* (PCNF), requiring all quantifiers to be in front of a purely propositional formula, which has to be in conjunctive normal form (CNF). However, natural encodings of problems from AI do not yield QBFs in such a normal form, and thus the particular instances have to be transformed. The transformation is performed in two steps, namely *prenexing* and transformation of the resulting purely propositional matrix into CNF. The drawbacks of this transformation are an increase in both formula size and variable number, or, even worse, the formula's structure is disrupted.

In this paper, we present a prover, qpro, which works on arbitrary QBFs. Its basic procedure is a generalized DPLL algorithm with enhanced dependency-directed backtracking techniques. The space requirements for qpro are modest; it runs in polynomial space (wrt the length of the input formula). The motivation to circumvent formulas

¹ This work was supported by FWF under grant P18019, ÖAD under grant Amadée 2/2006, and FFG under grant FIT-IT-810806.

² Institut für Informationssysteme 184/3, Technische Universität Wien, Favoritenstraße 9–11, A-1040 Vienna, Austria.

³ Institut für Softwaretechnik und Interaktive Systeme 188/3, Technische Universität Wien, Favoritenstraße 9–11, A-1040 Vienna, Austria.

in PCNF and to work with arbitrary QBFs is the problem to generate “good” normal forms in this logic. The main problem here is the handling of quantifiers at the places where they occur. This is in contrast to first-order logic. We explain in the following aspects of normalization for different logics and discuss why QBFs are problematic.

It is well known how a propositional (or first-order formula) can be translated into a satisfiability-equivalent CNF, such that the structural information is retained by new atoms [23, 10, 8]. Together with their definition, such new atoms can mimic the effect of the analytic cut rule in full calculi like Gentzen systems resulting in drastically shorter proofs [3, 11]. Moreover, as experiments showed [14, 21], such structure-preserving translations are not only beneficial from a theoretical point of view, but can also speed-up automated theorem provers for practical problems. In the last few years, similar results have been obtained for the case of prenex QBFs and an optimized handling of the newly introduced atoms has been proposed [1].

We consider the problem to construct a prenex form of a QBF. The prenexing transformation cannot be carried out deterministically; the chosen normalization strategy crucially influences the runtimes (also depending on the concrete solver used), see e.g., [15, 26]. In fact, this phenomenon mirrors a similar observation from classical theorem proving in *first-order logic*, where classes of formulas exist for which different quantifier shifting strategies (resulting in different prenex forms) yield a non-elementary difference of proof size (and search-space size) [4, 12]. Clearly, the impact of the prenex forms is less drastic for QBFs because of the simpler underlying logic, but there are indications that prenexing impacts the runtime of highly optimized state-of-the-art solvers [18].

In first-order logic, skolemization can be used to encode the properties of (usually) existential quantifiers by Skolem functions. In a nutshell, skolemization gets rid of existential quantifiers “in place”. The introduced Skolem functions encode two properties of quantifier rules in full first-order calculi: (i) the eigenvariable condition and (ii) non-permutabilities between quantifier rules. Condition (i) is satisfied by the requirement to introduce a globally new function symbol, and condition (ii) is handled by the occur check in the unification algorithm. Due to the weaker syntax of QBFs, the introduction of Skolem functions is not possible and therefore this conceptually simple tool is not directly applicable in the context of QBFs.

The outline of the paper is as follows: Section 2 introduces necessary definitions and notations. Sections 3 and 4 are devoted to the formal underpinnings of the prover. Section 5 provides experiments and comparisons. Finally, we discuss the results as well as related and future work in Section 6.

2 Background

We introduce the language \mathcal{L}_P of QBFs as an extension of the language of propositional logic. The alphabet of \mathcal{L}_P consists of paren-

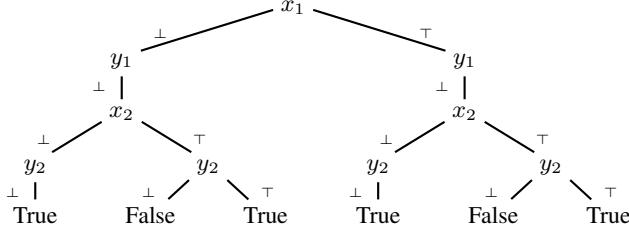


Figure 1. The branching tree of the example QBF ϕ .

theses, the truth constants \top and \perp , a countable set of variables \mathcal{P} , the unary connective \neg (negation), the binary connectives \vee (disjunction) and \wedge (conjunction), and the quantifier symbols \forall (universal) and \exists (existential). A literal is a variable or a negated variable.

We define the language of *quantified propositional logic* over a set of variables \mathcal{P} as the smallest set, $\mathcal{L}_{\mathcal{P}}$, satisfying the conditions:

1. If $x \in \mathcal{P} \cup \{\top, \perp\}$, then $x \in \mathcal{L}_{\mathcal{P}}$ and $(\neg x) \in \mathcal{L}_{\mathcal{P}}$;
2. if $\phi, \psi \in \mathcal{L}_{\mathcal{P}}$, then $(\phi \circ \psi) \in \mathcal{L}_{\mathcal{P}}$, where $\circ \in \{\vee, \wedge\}$;
3. if $\phi \in \mathcal{L}_{\mathcal{P}}$, $x \in \mathcal{P}$, and Qx does not occur in ϕ , then $(Qx \phi) \in \mathcal{L}_{\mathcal{P}}$, where $Q \in \{\forall, \exists\}$.

Any element of $\mathcal{L}_{\mathcal{P}}$ is called a *quantified boolean formula* (QBF). If no ambiguities arise, we omit parentheses when convenient. Moreover, for a set $X = \{x_1, \dots, x_n\}$ of variables, and $Q \in \{\forall, \exists\}$, we write $QX\phi$ or $Qx_1 \dots x_n \phi$ as a short-hand for $Qx_1 \dots Qx_n \phi$.

Note that we allow negation only in front of a variable or a truth constant. Formulas which obey this restriction are said to be in *negation normal form* (NNF). The NNF of any QBF can be obtained by applying DeMorgan's laws and the removal of double negation. Since this transformation can be done deterministically and since the increase of the formula size is negligible, we only consider QBFs in NNF. From a technical point of view, QBFs in NNF have the advantage that polarities of complex subformulas do not come into play.

The scope of a quantifier (occurrence) Qx in a QBF ϕ is defined as ψ , where $Qx \psi$ is the subformula corresponding to the occurrence of Qx in ϕ . An occurrence of a variable x is called existential (resp. universal) if it is located within the scope of a quantifier $\exists x$ (resp. $\forall x$). Unless stated otherwise, we consider *closed* QBFs, i.e., each occurrence of a variable x is located in the scope of a quantifier Qx .

We denote by $\phi[x/\psi]$ the result of substituting each occurrence of a variable x in a QBF ϕ by a QBF ψ . The semantics of a QBF is then given as follows: A QBF $\exists x \psi$ is true iff $\psi[x/\perp]$ or $\psi[x/\top]$ is true. Dually, a QBF $\forall x \psi$ is true iff $\psi[x/\perp]$ and $\psi[x/\top]$ are true. The other connectives are treated according to the standard evaluation rules of propositional logic, providing the usual recursive definition of the truth value of a QBF. Two closed QBFs are called equivalent if they possess the same truth value.

The sequence of the variable assignments when evaluating a QBF can be illustrated by a branching tree. The nodes contain the variables and the two subtrees of a node x correspond to the subproblems, where x is replaced by \perp or \top , as indicated by the labels of the arcs. The leaves contain the resulting truth values. If x is existentially (resp. universally) quantified and one subproblem evaluates to true (resp. false), then the second subproblem can be omitted. As an example, we show in Figure 1 the branching tree of the true formula

$$\phi = \forall x_1 \exists y_1 (\forall x_2 \exists y_2 ((x_2 \vee \neg y_2) \wedge (\neg x_2 \vee y_2)) \vee (x_1 \wedge y_1)).$$

Definition 1 A QBF ϕ is given in *prenex normal form* (PNF) if ϕ is of the form $Q_1 X_1 \dots Q_m X_m \psi$, where $Q_i \in \{\forall, \exists\}$ and ψ is purely

```
BOOLEAN split(QBF φ in NNF) {
    switch (simplify (φ)):
        case T: return True;
        case ⊥: return False;
        case (φ1 ∨ φ2): return (split(φ1) || split(φ2));
        case (φ1 ∧ φ2): return (split(φ1) && split(φ2));
        case (QXψ): select x ∈ X;
            if Q = ∃ return (split(ψ[x/T]) || split(ψ[x/⊥]));
            if Q = ∀ return (split(ψ[x/T]) && split(ψ[x/⊥]));
    }
}
```

Figure 2. The basic algorithm.

propositional. Moreover, if ψ is in conjunctive normal form, ϕ is said to be in *prenex conjunctive normal form* (PCNF).

The PCNF format is required by most of the available QBF solvers.

Any QBF can be translated into an equivalent QBF in PNF, but there are several ways to do this. The concept of different *prenexing strategies* is discussed in [15, 26]. We give here only some intuition.

First, the *dependencies* between the quantifiers in a QBF are given by common occurrences on paths in the formula tree. To avoid a formal definition, consider the following example

$$\psi = \exists x ((\forall y_1 \exists z_1 \forall u_1 \exists v_1 \psi_1) \wedge (\forall y_2 \exists z_2 \psi_2) \wedge (\exists y_3 \forall z_3 \psi_3)),$$

where the ψ_i 's are propositional formulas. Then, $\forall y_1$ depends on $\exists x$, $\exists z_1$ depends on $\forall y_1$ as well as on $\exists x$, $\forall y_2$ depends on $\exists x$, etc. but, e.g., $\exists z_2$ does not depend on $\forall y_1$. We say that a QBF has *depth m*, if the sequences of depending quantifiers provide at most $m-1$ alternations. Observe that ψ has depth 5 as witnessed by the “path” $\exists x \forall y_1 \exists z_1 \forall u_1 \exists v_1$. The aim of prenexing is to “linearize” quantifier dependencies (which in fact form a partial order) without increasing the depth of the QBF. We consider here four different *prenexing strategies*, namely “↑”, “↓”, “ $\exists \downarrow \forall \uparrow$ ”, and “ $\exists \uparrow \forall \downarrow$ ”. Hereby, “↑” (resp. “↓”) denotes that any quantifier is placed as outermost (resp. innermost) as possible in the prefix. “ $\exists \downarrow \forall \uparrow$ ” and “ $\exists \uparrow \forall \downarrow$ ” follow the same concept but now the handling is depending on the particular quantifier, i.e., whether it concerns an existential or a universal one. Thus, for our example formula ψ , we derive different PNFs of ψ having the same depth:

$$\begin{aligned} \uparrow &: \exists x y_3 \forall y_1 y_2 z_3 \exists z_1 z_2 \forall u_1 \exists v_1 (\psi_1 \wedge \psi_2 \wedge \psi_3); \\ \exists \downarrow \forall \downarrow &: \exists x y_3 \forall y_1 y_2 \exists z_1 z_2 \forall u_1 z_3 \exists v_1 (\psi_1 \wedge \psi_2 \wedge \psi_3); \\ \exists \downarrow \forall \uparrow &: \exists x \forall y_1 y_2 \exists z_1 y_3 \forall u_1 z_3 \exists v_1 z_2 (\psi_1 \wedge \psi_2 \wedge \psi_3); \\ \downarrow &: \exists x \forall y_1 \exists z_1 y_3 \forall u_1 y_2 z_3 \exists v_1 z_2 (\psi_1 \wedge \psi_2 \wedge \psi_3). \end{aligned}$$

QBFs in PNF are prototypical problems for complexity classes in the *polynomial hierarchy*. In fact, the evaluation problem of QBFs $\exists X_1 \forall X_2 \dots Q_i X_i \phi$ is Σ_i^P -complete with $Q_i = \exists$ if i is odd and $Q_i = \forall$ if i is even. Dually, evaluating $\forall X_1 \exists X_2 \dots Q_i X_i \phi$ is Π_i^P -complete, with $Q_i = \forall$ if i is odd, and $Q_i = \exists$ if i is even.

3 A Generalization of the DPLL Procedure

In this section we present the most important formal underpinnings of the presented solver qpro which relies on a generalized variant of the decision procedure due to Davis, Putnam, Loveland, and Logemann (DPLL for short). The DPLL procedure [9] represents one of the most successful algorithms to decide the truth value of a propositional formula. This method has been adapted for QBFs (in PCNF format) and it is used in many state-of-the-art solvers.

We generalize DPLL in such a way that it is applicable to arbitrary formulas in $\mathcal{L}_{\mathcal{P}}$, i.e., such that DPLL can be directly applied to QBFs

in NNF without additional transformations to PCNF. Figure 2 shows a simplified version of the program code of our decision procedure.

Note that DPLL is a direct implementation of the semantics for QBFs. The formula is split into subproblems, whose return values are treated according to the connective, the variables are replaced by truth constants, and simplifications are applied until a truth value is obtained. The basic decision procedure is thus a simple search-based backtracking algorithm which works in polynomial space with respect to the size of the input formula.

The function `simplify`(ϕ) in Figure 2 returns a formula which results from ϕ by applying numerous equivalence-preserving transformations, including, e.g., the following ones:

- (a) $\neg T \Rightarrow \perp$; $\neg \perp \Rightarrow T$;
- (b) $T \wedge \phi \Rightarrow \phi$; $\perp \wedge \phi \Rightarrow \perp$; $T \vee \phi \Rightarrow T$; $\perp \vee \phi \Rightarrow \phi$;
- (c) $(Qx \phi) \Rightarrow \phi$, $Q \in \{\forall, \exists\}$, x does not occur in ϕ ;
- (d) $\forall x(\phi \wedge \psi) \Rightarrow (\forall x\phi) \wedge (\forall x\psi)$;
- (e) $\forall x(\phi \vee \psi) \Rightarrow (\forall x\phi) \vee \psi$, whenever x does not occur in ψ ;
- (f) $\exists x(\phi \vee \psi) \Rightarrow (\exists x\phi) \vee (\exists x\psi)$;
- (g) $\exists x(\phi \wedge \psi) \Rightarrow (\exists x\phi) \wedge \psi$, whenever x does not occur in ψ .

Rewritings (d)–(g) are known as *miniscoping*. Note that the application of miniscoping is dynamic within the algorithm, due to the repetitive substitution of variables and simplifications of subformulas. Further simplifications are derived from generalizations of other well known concepts.

Definition 2 Let ϕ be a QBF, ψ a subformula of ϕ , $Q \in \{\forall, \exists\}$, and $\circ \in \{\vee, \wedge\}$. A literal $l \in \{x, \neg x\}$ is called

- local unit (wrt ψ) in ϕ , if ψ is of the form $(l \circ \psi')$;
- global unit (wrt ψ) in ϕ , if ψ is of the form $Qx(l \circ \psi')$;
- pure (wrt ψ) in ϕ , if ψ is of the form $Qx\psi'$, and \bar{l} does not occur in ψ' , where $\bar{x} = \neg x$ and $\bar{\neg x} = x$.

Proposition 1 Let ϕ be a closed QBF and let $l = x$ (resp. $l = \neg x$) be a (i) local-unit (ii) global-unit (iii) pure literal wrt a subformula ψ in ϕ , where ψ is given according to Definition 2. Then, ϕ is equivalent to the QBF resulting from ϕ by replacing ψ in case of

- (i) by $\begin{cases} l \circ \psi'[x/\top] & (\text{resp. } l \circ \psi'[x/\perp]) \\ l \circ \psi'[x/\perp] & (\text{resp. } l \circ \psi'[x/\top]) \end{cases}$ if $\circ = \wedge$;
- (ii) by $\begin{cases} (l \circ \psi')[x/\top] & (\text{resp. } (l \circ \psi')[x/\perp]) \\ (l \circ \psi')[x/\perp] & (\text{resp. } (l \circ \psi')[x/\top]) \end{cases}$ if x is existential;
- (iii) by $\begin{cases} \psi'[x/\top] & (\text{resp. } \psi'[x/\perp]) \\ \psi'[x/\perp] & (\text{resp. } \psi'[x/\top]) \end{cases}$ if x is universal.

Note that (ii) and (iii) delete Qx in ψ by an implicit application of (c) (since all occurrences of x in the scope of Qx have been replaced by \perp or \top). Also observe the following difference between (i) and (ii). In (i), the literal occurrence l in $(l \circ \psi)$ is not affected by the substitution of x , while in (ii) the substitution of x concerns also l in $(l \circ \psi)$. Note that (ii) yields—together with (a) and (b)—a replacement of the entire subformula $\psi = Qx(l \circ \psi')$ in ϕ by \top (resp. \perp), in the case $Q = \exists, \circ = \vee$ (resp. $Q = \forall, \circ = \wedge$). Finally, observe that (ii) also applies to formulas of the form $\psi = Qx_1 Q_1 X_1 \dots Q_n X_n (l \circ \psi'')$ since, by definition of QBFs, $x \notin X_i$, and with (e) and (g), we can transform ψ to be of form $Qx(l \circ \psi'')$.

The remaining part of the extended DPLL procedure in Figure 2 is straightforward but we have to face three sources of indeterminism

within the switch statement: if `simplify`(ϕ) returns a QBF $\phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$, we have to select (i) which subformula to evaluate first; this part differs from PCNF solvers, where such a decision is not necessary; if `simplify`(ϕ) returns $QX\psi$, with $Q \in \{\forall, \exists\}$, we have to select (ii) on which variable $x \in X$ to branch, and (iii) which subproblem (i.e., $\psi[x/\top]$ or $\psi[x/\perp]$) to consider first. Choice (ii) is obviously restricted by the dependencies between the quantifiers, since universal and existential quantifiers must not be permuted.

4 Dependency-Directed Backtracking

In this section we briefly describe an important technique called *dependency-directed backtracking* (DDB), which is known to be crucial for the efficiency of the DPLL procedure. In QBF solvers for PCNF, this technique only works for false subproblems. Starting from [20], we generalize this technique to arbitrary formulas, for which it can be applied to *true and false subproblems*. DDB for false subproblems applies to existential variables, whereas DDB for true subproblems applies to universal variables.

When we inspect a branching tree, we may notice that some branches (together with the corresponding subtrees) can be omitted without influencing the (partial) evaluation result. In this case, the result is independent from the assignments of some variables. If we have set such a variable x to one truth value, we can safely omit the assignment of x to the other truth value at this point in the branching tree during backtracking. Consider Figure 1 as an example and observe that the tree is *symmetric*, i.e., the left and the right subtree of x_1 (both with root y_1) are identical. Suppose the prover has finished the left half of the branching tree. Under the assignment \perp for x_1 , the resulting simplified formula $\phi' : \forall x_2 \exists y_2 ((x_2 \vee \neg y_2) \wedge (\neg x_2 \vee y_2))$ evaluates to true. Since x_1 is a universal variable, the next assignment for it is \top . A clever solver notices that x_1 has no influence on the truth value of ϕ' with the consequence that ϕ' is true under the new assignment. So, no further assignments are performed because we can utilize the result of the left subtree with root y_1 . What we have described here is DDB on true subproblems (neglecting substitutions of unit and pure literals, however, for the matter of presentation). The more common DDB on false subproblems works as expected.

We have implemented two different sound and complete backtracking techniques, namely *labeling* and *relevance sets*. Since the latter is superior to the former and since all experiments have been performed using the latter, we only describe *DDB by relevance sets* here. We present this technique only for true subproblems, but it works dually for false ones.

Let R_y denote the *relevance set* for a node y . If the solver reaches a leaf l in the branching tree, only those variables whose assignments determine the truth value of the formula (i.e., the label of l) form R_l . Assume we return from a subtree P_1 with root x_1 to the variable x directly above x_1 during backtracking and P_1 is true. If x is existential, then $R_x = R_{x_1}$. If x is universal, then we check whether x is contained in R_{x_1} . If $x \notin R_{x_1}$ holds, then the other subproblem P'_1 is true and $R_x = R_{x_1}$. Otherwise, the relevance set R'_{x_1} of P'_1 has to be considered. We construct R_x as follows. If $x \notin R'_{x_1}$ holds, then $R_x = R'_{x_1}$. Otherwise, R is set to $R_{x_1} \cup R'_{x_1}$.

5 Experimental Evaluation

In this section we compare the performance of our new solver `qpro` against the established systems QuBE-BJ [17] (v1.2), `sKizzo` [5] (v0.4), `semprop` [20] (rel. 24/02/02), and `quantor` [7] (rel. 25/01/04). These solvers have been selected because they have shown to be

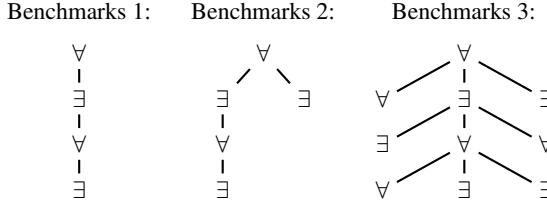


Figure 3. Quantifier dependencies of the different benchmarks.

competitive in previous QBF evaluations and do not deliver wrong results on our benchmarks. Moreover, QuBE–BJ and semprop implement backtracking techniques, similar to the one used in qpro. Finally, sKizzo and quantor try to extract original quantifier dependencies from a PCNF, and thus may detect similar structural information on the input formula as qpro has got *a priori* from the input of the corresponding nonprenex formula.

All solvers except qpro require the input to be in PCNF. We thus apply the following test strategy: Given a benchmark QBF ϕ not in PCNF, we (i) provide ϕ as input to qpro; (ii) translate ϕ into PCNF and provide the outcome as input to the other solvers. The exact way we obtained the PCNFs is discussed below. We have chosen the following benchmark formulas (not in PCNF).⁴

1. Encodings for satisfiability in modal logic K .
2. Encodings for correspondence tests in answer-set programming.
3. Encodings of reasoning with nested counterfactuals.

In what follows, we briefly describe the benchmarks. For detailed information, the reader is referred to the according references.

The first set of benchmarks contains instances which were also used in the TANCS’98 comparison of provers for modal logics. Applying the encoding from [22] yields QBFs with a linear dependency among the quantifiers. Hence, the translation to PNF is fully determined (there is just one way to shift the quantifiers in front of the formula).

The benchmarks in the second set encode correspondence tests between propositional logic programs under the answer-set semantics, where it is checked whether two programs provide equal projected answer-sets under any program extension over a specified alphabet, cf. [25]. For the benchmarks, we use two strategies, namely “ \uparrow ” and “ \downarrow ” to obtain formulas in PCNF.

The benchmarks in the third set encode the problem of reasoning over nested counterfactuals [15]. For this set of formulas, the quantifier dependencies allow for several different translations into PCNF. We have applied each strategy from [15] to the PCNF solvers. To be fair, we show only the results for the best strategy of each solver.

The structural differences between these sets are best illustrated by the quantifier dependencies of the formulas. We depict them for the case of QBFs of depth 4 in Figure 3. Hence, the chosen benchmarks provide an increasing complexity in their structure, and therefore, an increasing disruption of the structure during the transformation to PNF can be expected. In particular, for Benchmarks 1, we just can investigate the effect of applying the transformation of one shifting strategy, since the prefix is already determined by the encoding. Benchmarks 2 allow to analyze the effect of prenexing if there is only a small deviation from a linear quantifier dependency. In fact, using either strategy “ \uparrow ” or “ \downarrow ”, one can place the group of existential quantifiers from the right path either together with the upper or the lower set of existential quantifiers from the left path. Finally, Benchmarks 3 provide formulas, for which different PNFs can be obtained.

⁴ PCNF versions are part of the QBFLIB, <http://www.qbflib.org/>.

	number of timeouts					average runtimes (in sec.)				
	qpro	QuBE–BJ	semprop	sKizzo	quantor	qpro	QuBE–BJ	semprop	sKizzo	quantor
01-*	10	7	7	0	18	52.34	37.43	38.29	12.65	67.90
02-*	2	0	0	0	18	23.90	6.92	0.08	0.89	71.67
03-*	14	10	14	17	17	70.30	52.92	68.27	80.72	80.69
04-*	9	8	0	0	17	45.06	45.22	0.01	0.18	79.79
05-*	0	0	0	7	13	0.01	0.00	0.11	37.25	63.09
06-*	0	0	0	0	12	0.00	0.00	0.32	0.29	54.90
07-*	0	0	0	0	8	0.00	0.00	0.00	0.17	43.10
08-*	0	0	0	4	8	0.00	0.00	0.01	0.12	40.24
09-*	0	0	0	1	2	0.00	0.00	0.01	2.33	15.09
10-*	0	0	0	0	0	0.00	0.00	0.01	0.00	0.07
11-*	9	8	0	0	16	45.09	43.63	0.14	0.22	76.02
12-*	6	6	0	0	15	34.28	32.67	0.07	0.21	72.10
13-*	0	4	0	0	1	0.22	26.74	0.02	1.00	6.79
14-*	13	13	12	10	11	72.26	64.57	58.89	50.13	56.22
15-*	0	0	0	0	0	3.90	0.01	0.39	0.04	0.05
16-*	0	0	0	0	19	0.27	0.28	0.01	0.57	73.96
17-*	16	0	12	4	20	78.74	0.72	61.29	47.83	82.57
18-*	13	0	0	6	18	65.63	0.26	0.04	41.46	79.95

Table 1. Benchmarks 1: Modal Logic K.

	number of timeouts					average runtimes (in sec.)				
	qpro	QuBE–BJ	semprop	sKizzo	quantor	qpro	QuBE–BJ	semprop	sKizzo	quantor
S \uparrow	—	842	117	527	1000	—	87.34	81.82	74.67	100
S \downarrow	—	90	6	0	1000	—	43.21	27.60	2.67	100
T \uparrow	—	43	38	0	1000	—	20.85	54.86	2.97	100
T \downarrow	—	0	0	0	1000	—	9.26	16.90	1.13	100
S	29	—	—	—	—	33.37	—	—	—	—
T	0	—	—	—	—	17.87	—	—	—	—

Table 2. Benchmarks 2: Answer Set Correspondence.

We ran our tests on an Intel Xeon 3 GHz with 4GB of RAM. All solvers are used with their predefined standard options. For each instance, we set a timeout of 100 seconds. We report both the number of timeouts per set of benchmarks, as well as the average runtimes, where formulas with timeout are considered to be solved in 100 seconds. In what follows, we present our results on the benchmarks.⁵

Benchmarks 1: Modal Logic K. This set contains 378 formulas arranged in 18 subsets, with 21 formulas each. Half of the formulas evaluates to true. Depending on the modal depth of the original formula, the depth of the encodings ranges from 5 to 133; the number of variables ranges from less than 40 to more than 4300. Due to the transformation into PCNF, the number of variables increases up to more than 12800 in the worst case. The results are given in Table 1.

Benchmarks 2: Answer-Set Correspondence. Here, the test series comprise 1000 instances (465 are true and 535 are false). We ran each problem on two different encodings, S and T, where T is an explicit optimization of S (see [25] for details). The problem of answer-set correspondence is Π_4^P -complete, and thus all QBFs in this set have depth 4. The quantifier dependencies, as depicted in Figure 3, are the same for S and T and suggest to distinct between two strategies for obtaining PCNFs, viz. \uparrow and \downarrow . The QBFs possess, in case of S, 200 variables and, in case of T, 152 variables. The additional translation into PCNF yields, in case of S, QBFs over 2851 variables and, in case of T, QBFs over 2555 variables. The results for each solver on each combination of the chosen translation and (in the case of PCNF solvers) prenexing strategy are given in Table 2.

Benchmarks 3: Nested Counterfactuals. The final set of benchmarks are encodings of nested counterfactual reasoning, separated by

⁵ We highlight the best runtime for each test set (discrepancies due to measurement inaccuracy are ignored).

the depth of the resulting QBFs which ranges from 4 to 8. For each depth, we created 50 instances, where the QBFs contain 183, 245, 309, 375, and 443 variables. The transformation to PCNF increases the number of variables to 464, 600, 786, 934, and 1132. In total, we have about 60% true and 40% false instances. As mentioned above, these encodings allow for several different prenexing strategies. For space reasons, we do not present the results for each strategy here, but Table 3 shows the results for the *best* strategy which has been derived for each solver on the entire set of QBFs. Recall that for qpro we do not need to apply any such strategy.

		number of timeouts				average runtimes (in sec.)			
		qpro	QuBE-BJ	semprop	sKizzo	quantor	qpro	QuBE-BJ	semprop
		↑	↑	↑	↑	↑	↑	↑	↑
4	0	1	8	9	31	0.41	5.10	39.39	22.30
5	0	3	10	13	30	1.06	9.35	28.69	32.72
6	0	4	34	26	42	2.06	11.66	69.01	57.20
7	0	8	32	28	41	2.34	20.45	63.34	60.06
8	0	12	45	38	41	6.81	32.08	79.72	78.55
									90.47

Table 3. Benchmarks 3: Nested Counterfactuals.

6 Discussion and Conclusion

We presented a new QBF solver, qpro, which significantly differs from previous approaches by its ability to process arbitrary QBFs instead of QBFs in PCNF. We sketched generalizations of the DPLL procedure necessary to handle arbitrary QBFs and briefly discussed implemented performance-improving techniques like different forms of dependency-directed backtracking. Future work calls for an analysis how the different pruning techniques influence performance.

In practical applications, QBF solvers can be used as a black box in reasoning systems to solve encodings of the problems considered. Usually such encodings are not in PCNF and, as we have shown in our experiments, avoiding the additional translation into PCNF may result in much better performance. In what follows, we briefly discuss two main observations from our experiments.

- The more information on quantifier dependencies is lost due to prenexing, the more competitive qpro turns out to be. Table 3 contains those benchmarks where this effect is most apparent. Although we compare qpro here against the PCNF solvers together with their *best* suited strategy, qpro significantly outperforms all other solvers.
- Table 2 presents results for two different encodings of the same problem where T is an explicit optimization of S. These results show that qpro is less depending on the chosen encoding, whereas the performance of PCNF solvers differs much more. In fact, qpro performs better on the unoptimized encoding S, in the case the “wrong” prenexing strategy “↑” is used for the PCNF solvers.

Finally, we briefly discuss related systems.

- There are a few further solvers, namely QUBOS [2], boole⁶, and zqsat [16], which also allow arbitrary QBFs as input, but rely on different techniques. QUBOS simplifies the QBF and then constructs an equivalent propositional formula which is evaluated by SAT solvers, whereas boole is based on binary decision diagrams (BDDs). Thus both need exponential space in the worst case. We have included boole in our pre-tests, but it was not competitive at the benchmarks. We also neglected QUBOS, because it yielded wrong results on some problems. Finally, zqsat implements DPLL using zero-compressed BDDs. The comparison to zqsat is subject to future work.
- In [18], an extension to QuBE was suggested, where the input is a QBF in PCNF together with information on quantifier dependencies. Contrary to this approach, ours avoids the bounded renaming of

⁶ <http://www.cs.cmu.edu/~modelcheck/bdd.html>.

variables during the prefix construction together with the necessity to transform the matrix into CNF. This enables us to dynamically (i) apply miniscoping and (ii) recognize independent subproblems which can be solved in parallel. Nonetheless, the results in [18] lead to observations in the same direction as ours. Future work will include a detailed comparison of the two approaches, which will be done as soon as the modified QuBE system is available.

References

- [1] C. Ansótegui, C. Gomes, and B. Selman, ‘The Achilles’ Heel of QBF’, in *Proc. AAAI’05*, pp. 275–281. AAAI Press, (2005).
- [2] A. Ayari and D. Basin, ‘QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers’, in *Proc. FMCAD’02*, pp. 187–201. Springer, (2002).
- [3] M. Baaz, C. Fermüller, and A. Leitsch, ‘A Non-Elementary Speed Up in Proof Length by Structural Clause Form Transformation’, in *Proc. LICS’94*, pp. 213–219. IEEE Computer Society Press, (1994).
- [4] M. Baaz and A. Leitsch, ‘On Skolemization and Proof Complexity’, *Fundamenta Informaticae*, **20**, 353–379, (1994).
- [5] M. Benedetti, ‘sKizzo: A Suite to Evaluate and Certify QBFs’, in *Proc. CADE-21*, pp. 369–376. Springer, (2005).
- [6] P. Besnard, T. Schaub, H. Tompits, and S. Woltran, ‘Representing Paraconsistent Reasoning via Quantified Propositional Logic’, in *Inconsistency Tolerance*, 84–118, Springer, (2005).
- [7] A. Biere, ‘Resolve and Expand’, in *Proc. SAT’04*, pp. 59–70. Springer, (2005).
- [8] T. Boy de la Tour, ‘An Optimality Result for Clause Form Translation’, *Journal of Symbolic Computation*, **14**(4), 283–302, (1992).
- [9] M. Davis, G. Logemann, and D. Loveland, ‘A Machine Program for Theorem Proving’, *Comm. of the ACM*, **5**(7), 394–397, (1962).
- [10] E. Eder, *Relative Complexities of First-Order Calculi*, Vieweg, 1992.
- [11] U. Egly, ‘On Different Structure-preserving Translations to Normal Form’, *Journal of Symbolic Computation*, **22**, 121–142, (1996).
- [12] U. Egly, ‘Quantifiers and the System KE: Some Surprising Results’, in *Proc. CSL’98*, pp. 90–104. Springer, (1999).
- [13] U. Egly, T. Eiter, H. Tompits, and S. Woltran, ‘Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas’, in *Proc. AAAI’00*, pp. 417–422, (2000).
- [14] U. Egly and T. Rath, ‘On the Practical Value of Different Definitional Translations to Normal Form’, in *Proc. CADE-13*, pp. 403–417. Springer, (1996).
- [15] U. Egly, M. Seidl, H. Tompits, S. Woltran, and M. Zolda, ‘Comparing Different Prenexing Strategies for Quantified Boolean Formulas’, in *Proc. SAT’03*, pp. 214–228, (2004).
- [16] M. Ghasemzadeh, V. Klotz, and C. Meinel, ‘Embedding Memoization to the Semantic Tree Search for Deciding QBFs’, in *Proc. AI’05*, pp. 681–693. Springer, (2004).
- [17] E. Giunchiglia, M. Narizzano, and A. Tacchella, ‘Backjumping for Quantified Boolean Logic Satisfiability’, *AII*, **145**, 99–120, (2003).
- [18] E. Giunchiglia, M. Narizzano, and A. Tacchella, ‘Quantifier Structure in Search Based Procedures for QBFs’, in *Proc. of DATE*, (2006).
- [19] D. Le Berre, M. Narizzano, L. Simon, and A. Tacchella, ‘The Second QBF Solvers Comparative Evaluation’, in *Proc. SAT’04*, pp. 376–392. Springer, (2005).
- [20] R. Letz, ‘Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas’, in *Proc. TABLEAUX’02*, pp. 160–175. Springer, (2002).
- [21] A. Nonnengart, G. Rock, and C. Weidenbach, ‘On Generating Small Clause Normal Forms’, in *Proc. CADE-15*, pp. 397–411. Springer, (1998).
- [22] G. Pan and M. Vardi, ‘Optimizing a BDD-Based Modal Solver’, in *Proc. CADE-19*, pp. 75–89. Springer, (2003).
- [23] D. A. Plaisted and S. Greenbaum, ‘A Structure Preserving Clause Form Translation’, *Journal of Symbolic Computation*, **2**(3), 293–304, (1986).
- [24] J. Rintanen, ‘Constructing Conditional Plans by a Theorem Prover’, *Journal of Artificial Intelligence Research*, **10**, 323–352, (1999).
- [25] H. Tompits and S. Woltran, ‘Towards Implementations for Advanced Equivalence Checking in Answer-Set Programming’, in *Proc. ICLP’05*, pp. 189–203. Springer, (2005).
- [26] Michael Zolda, *Comparing Different Prenexing Strategies for Quantified Boolean Formulas*, Master’s thesis, TU Wien, 2004.

Knowledge Engineering for Bayesian Networks: How Common Are Noisy-MAX Distributions in Practice?

Adam Zagorecki¹ and Marek Druzdzel²

Abstract. One problem faced in knowledge engineering for Bayesian networks is the exponential growth of the number of parameters in their conditional probability tables (CPTs). The most common practical solution is application of the noisy-OR (or their generalization, the noisy-MAX) gates, which take advantage of independence of causal interactions and provide a logarithmic reduction of the number of parameters required to specify a CPT. In this paper, we propose an algorithm that fits a noisy-MAX distribution to an existing CPT and we apply it to search for noisy-MAX gates in three existing practical Bayesian networks. We show that noisy-MAX gate provides a surprisingly good fit for as many as 50% of CPTs in these networks. The importance of this finding is that it provides an empirical justification for the use of the noisy-MAX gate as a powerful knowledge engineering tool.

1 INTRODUCTION

Bayesian networks [1] provide a convenient and sound framework for encoding uncertain knowledge and for reasoning with it. A Bayesian network (BN) is essentially an acyclic directed graph encoding a factorization of a joint probability distribution. The structure of the graph represents the variables and independencies among them, while the probability distributions over the individual variables conditional on their direct predecessors (parents) represent individual components of the factorization.

When a node of a BN and all its parents are discrete, the conditional probability distributions are stored in *conditional probability tables* (CPTs) indexed by all possible combinations of states of the parents. This poses considerable difficulties to knowledge engineering, to learning BNs from data, and to inference algorithms for BNs. An ingenious practical solution to this problem has been the application of parametric conditional distributions, such as the noisy-OR. By taking advantage of *independence of causal interaction* (ICI), these gates offer a reduction of the number of parameters required to specify a conditional probability distribution from exponential to linear in the number of parents.

The two most widely applied ICI distributions are the binary noisy-OR model [2] and its extension to multi-valued variables, the noisy-MAX model [3, 4]. Noisy-OR and noisy-MAX gates have proven their worth in many real-life applications (e.g., [5, 6, 7]). Their foremost advantage is a small number of parameters that are sufficient to specify the entire CPT. This leads to a significant reduction of effort in knowledge elicitation from experts [3, 5], improves

the quality of distributions learned from data [8], and reduces the spatial and temporal complexity of algorithms for Bayesian networks [9, 10].

Our research aims at better understanding of the applicability of the noisy-MAX gates in practical BN models. We achieve this by developing a technique for fitting noisy-MAX relationships to existing, fully specified CPTs. Having this technique, we can examine CPTs in existing practical BN models for whether they can be approximated by the noisy-MAX model.

We analyze CPTs in three existing sizeable Bayesian network models: ALARM [11], HAILFINDER [12] and HEPAR II [8]. We show that the noisy-MAX gate provides a surprisingly good fit for a significant percentage of distributions in these networks. We observed this in both, distributions elicited from experts and those learned from data and for two measures of distance between distributions. We test the robustness of this result by fitting the noisy-MAX distribution to randomly generated CPTs and observe that the fit in this case is poor. Obtaining a randomly generated CPT that can be reasonably approximated by a noisy-MAX gate is extremely unlikely, which leads us to the conclusion that our results are not a coincidence. We investigate the influence of the conversion to the noisy-MAX on the precision of posterior probability distributions.

We envision two applications of this technique. The first is a possible refocusing of knowledge engineering effort from obtaining an exponential number of numerical probabilities to a much smaller number of noisy-MAX parameters. While a parametric distribution may be only an approximation to a set of general conditional probability distributions, the precision that goes with the latter is often only theoretical. In practice, obtaining large numbers numerical probabilities is likely to lead to expert exhaustion and result in poor quality estimates. Focusing the expert's effort on a small number of parameters of a corresponding parametric distribution should lead to a better quality model. The second application of our technique is in approximate algorithms for Bayesian networks. Whenever the fit is good, a CPT can be replaced by an ICI gate, leading to potentially significant savings in computation [9].

The remainder of this paper is organized as follows. Section 2 introduces the noisy-OR and the noisy-MAX gates. Section 3 proposes our algorithm for fitting noisy-MAX parameters to an arbitrary CPT. Section 4 presents the empirical results of the experiments testing the goodness of fit for several existing practical networks. In Section 5, we propose an explanation of the observed results.

2 NOISY-OR AND NOISY-MAX

We will represent random variables by upper-case letters (e.g., X) and their values by indexed lower-case letters, (e.g., x_1). We use

¹ Defence Academy of the UK, Cranfield University, UK, email: zagorecki@gmail.com

² University of Pittsburgh, School of Information Sciences, Decision Systems Laboratory, USA, email: marek@sis.pitt.edu

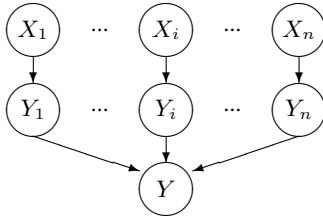


Figure 1. Direct modelling of noisy-OR

$Rng(X)$ to denote the range (the set of possible values) of a variable X . We assume that all variables are discrete. Let there be n binary nodes X_1, \dots, X_n , each with values from $Rng(X_i) = \{x_i, \bar{x}_i\}$. Let the variables X_i be the parents of an effect variable Y that assumes values y and \bar{y} .

A useful concept for modeling the noisy-OR by means of the deterministic OR, is that of the *inhibitor* nodes [1, 10], which model individual probabilistic relations between each cause and the effect individually. The general model including inhibitor nodes is shown in Fig. 1. The CPT of Y defines how those individual effects combine to produce Y . For the noisy-OR model, the CPT of node Y is equivalent to a deterministic OR. The *inhibitor* nodes Y_i introduce *noise* — the probabilistic effect of X_i on Y . The CPT of each Y_i is of the form: $\Pr(y_i|x_i) = p_i$ where $p_i \in [0, 1]$, and $\Pr(y_i|\bar{x}_i) = 0$. The noisy-MAX [3, 4] is basically an extension of the noisy-OR model to multi-valued variables. The noisy-MAX assumes that the variable Y has n_y states and that these states are ordered. The inhibitor nodes Y_i take values from the same domain as Y and their states follow the same ordering. Every parent variable X_i has n_i values. We use q_{ijk} to denote the element of CPT of inhibitor node Y_i that corresponds to the j -th value of parent node X_i and k -th value of Y : $\forall_{ijk} q_{ijk} = \Pr(y_i^k|x_i^j)$. Probabilities q_{ijk} are noisy-MAX parameters. The inhibitor variables Y_i have the same range as Y and their CPTs are constrained in the following way:

$$q_{ijk} = \begin{cases} 1 & \text{if } j = 1, k = 1 \\ 0 & \text{if } j = 1, k \neq 1 \\ p \in [0 \dots 1] & \text{if } j \neq 1 \end{cases}$$

The CPT of Y is a deterministic MAX defined by the ordering relation over states of Y . It is a common practice to add a *leak* term to the noisy-OR model. The leak is an auxiliary cause that serves the purpose of modeling the influence of causes that are not explicitly included in the model. In our experiments, we always assume that the noisy-MAX model includes the leak probability.

3 CONVERTING CPT INTO NOISY-MAX GATE

In this section, we propose an algorithm that fits a noisy-MAX distribution to an arbitrary CPT. In other words, it identifies the set of noisy-MAX parameters that produces a CPT that is the *closest* to the original CPT. Let C_Y be the CPT of a node Y that has n parent variables X_1, \dots, X_n . Each variable X_i can take one of n_{X_i} possible values. We use \mathbf{p}_i to denote i -th combination of the parents of Y and \mathbf{P} to denote the set of all combinations of parents values, $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_m\}$, where m is the product of the numbers of possible values of the X_i s, i.e., $m = \prod_{i=1}^n n_{X_i}$. There exist several measures of similarity of two probability distributions of which two are commonly used: Euclidean distance and Kullback-Leibler (KL) divergence. The main difference between the two is that the Euclidean

distance is based on absolute differences and, hence, is insensitive to large relative differences in very small probabilities, which is its major drawback. The KL divergence addresses this problem, but on the other hand, the problem with KL divergence is that for cases when the estimated probability is zero and the goal probability is non-zero, it is infinity. In our experiments, in such cases we replaced zero with a constant close to zero, which is a common practice.

The problem of defining a distance between two CPTs is somewhat more complicated, because a CPT is a set of probability distributions. The easiest approach is to define distance between two CPTs as a sum of distances of corresponding probability distributions in both CPTs. However, in practice not all distributions in the CPT are equally important. This is because typically some of the configurations of parents' states are far more likely than the others.

Definition 1 (Euclidean distance between two CPTs) *The distance D_E between two CPTs, $\Pr_A(Y|\mathbf{P})$ and $\Pr_B(Y|\mathbf{P})$ is a weighted sum of Euclidean distances between their corresponding probability distributions:*

$$\begin{aligned} D_E(\Pr_A(Y|\mathbf{P}), \Pr_B(Y|\mathbf{P})) &= \\ &= \sum_{i=1}^m w_{\mathbf{p}_i} \sum_{j=1}^{n_Y} \left(\Pr_A(y_j|\mathbf{p}_i) - \Pr_B(y_j|\mathbf{p}_i) \right)^2. \end{aligned} \quad (1)$$

Definition 2 (KL divergence between CPTs) *The divergence D_{KL} between the goal CPT $\Pr_A(Y|\mathbf{P})$ and its approximation $\Pr_B(Y|\mathbf{P})$ is a weighted sum of KL divergences between their corresponding probability distributions:*

$$\begin{aligned} D_{KL}(\Pr_A(Y|\mathbf{P}), \Pr_B(Y|\mathbf{P})) &= \\ &= \sum_{i=1}^m w_{\mathbf{p}_i} \sum_{j=1}^{n_Y} \Pr(y_j|\mathbf{p}_i) \ln \frac{\Pr_A(y_j|\mathbf{p}_i)}{\Pr_B(y_j|\mathbf{p}_i)}, \end{aligned} \quad (2)$$

In both definitions $w_{\mathbf{p}_i}$ is a weighting constant for each distribution in the CPT and $w_{\mathbf{p}_i} = P(\mathbf{p}_i)$.

Definition 3 (MAX-based CPT) *A MAX-based CPT $\Pr_q(Y|\mathbf{P})$ is a CPT constructed from a set of noisy-MAX parameters \mathbf{q} .*

Our goal is to find for a given fully specified CPT $\Pr_{cpt}(Y|\mathbf{P})$, such set of the noisy-MAX parameters \mathbf{q} that minimizes the Euclidean distance between the original CPT and the MAX-based CPT $\Pr_q(Y|\mathbf{p}_i)$. For simplicity, we will use θ_{ij} to denote the element of CPT that corresponds to the i -th element of \mathbf{P} and j -th state of Y . When this parameter is given in a fully defined CPT, we use upper index θ_{ij}^{cpt} , and when the parameter was obtained from the MAX-based CPT, we use upper index θ_{ij}^{max} . We can now rewrite Eq. 1 as:

$$\sum_{i=1}^m w_{\mathbf{p}_i} \sum_{j=1}^{n_Y} (\theta_{ij}^{cpt} - \theta_{ij}^{max})^2.$$

Because further part of our discussion relies heavily on cumulative probabilities, we introduce cumulative probability distributions based on the parameters θ_{ij} and q_{ij} . We define Θ_{ij} as:

$$\Theta_{ij} = \begin{cases} \sum_{k=1}^j \theta_{ik} & \text{if } j \neq 0 \\ 0 & \text{if } j = 0, \end{cases}$$

which constructs a cumulative probability distribution function for $\Pr(Y|\mathbf{p}_i)$. It is easy to notice, that $\theta_{ij} = \Theta_{ij} - \Theta_{i(j-1)}$. The next step is to express the MAX-based CPT parameters θ_{ij}^{max} in terms of the noisy-MAX parameters. In similar manner, we define the cumulative probability distribution of noisy-MAX parameters as:

$$Q_{ijk} = \begin{cases} \sum_{l=1}^k q_{ijl} & \text{if } j \neq 0 \\ 0 & \text{if } j = 0. \end{cases}$$

Pradhan et al. [6] proposed an algorithm exploiting cumulative probability distributions for efficient calculation of the MAX-based CPT that computes parameters of the MAX-based CPT as follows:

$$\Theta_{ij}^{max} = \prod_{x_p^r \in \mathbf{p}_i} Q_{prj}. \quad (3)$$

The product in Eq. 3 is taken over all elements of the cumulative distributions of noisy-MAX parameters, such that the values of a parent node X_i belong to a combination of parent states in CPT. Eq. 4 shows how to compute the element θ_{ij}^{max} from the noisy-MAX parameters:

$$\begin{aligned} \theta_{ij}^{max} &= \Theta_{ij}^{max} - \Theta_{i(j-1)}^{max} \\ &= \prod_{x_p^r \in \mathbf{p}_i} Q_{prj} - \prod_{x_p^r \in \mathbf{p}_i} Q_{pr(j-1)} \\ &= \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^j q_{prk} - \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^{j-1} q_{prk}. \end{aligned} \quad (4)$$

However, parameters θ_{ij}^{max} have to obey the axioms of probability, which means that we have only $n_Y - 1$ independent terms and not n_Y , as the notation suggests. Hence, we can express θ_{ij}^{max} in the following way:

$$\theta_{ij}^{max} = \begin{cases} \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^j q_{prk} - \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^{j-1} q_{prk} & \text{if } j \neq n_Y \\ 1 - \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^{n_Y-1} q_{prk} & \text{if } j = n_Y. \end{cases}$$

Theorem 1 *The distance D_E between an arbitrary CPT $\Pr_{cpt}(Y|\mathbf{P})$ and a MAX-based CPT $\Pr_q(Y|\mathbf{P})$ of noisy-MAX parameters \mathbf{q} as a function \mathbf{q} has exactly one minimum.*

Proof 1 *We prove that for each noisy-MAX parameter $q \in \mathbf{q}$, the first derivative of D_E has exactly one zero point. We will subsequently show that the second derivative is always positive, which indicates that D_E has exactly one minimum and proves the theorem. The first derivative of D_E over q is*

$$\begin{aligned} \frac{\partial}{\partial q} \sum_{i=1}^m w_{\mathbf{p}_i} \sum_{j=1}^{n_Y-1} \left(\theta_{ij}^{cpt} - \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^j q_{prk} + \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^{j-1} q_{prk} \right)^2 \\ + \frac{\partial}{\partial q} w_{\mathbf{p}_i} \sum_{i=1}^m \left(- \sum_{j=1}^{n_Y-1} \theta_{ij}^{cpt} + \prod_{x_p^r \in \mathbf{p}_i} \sum_{k=1}^{n_Y-1} q_{prk} \right)^2. \end{aligned}$$

Each of the three products contains at most one term q and, hence, the expression takes the following form:

$$\frac{\partial}{\partial q} \sum_{i,j} (A_{ij} + B_{ij}q)^2, \quad (5)$$

where A_{ij} and B_{ij} are constants. At least some of the terms B_{ij} have to be non-zero (because external sum in Eq. 5 runs over all elements of the CPT). The derivative

$$\frac{\partial}{\partial q} \sum_{i,j} (A_{ij} + B_{ij}q)^2 = 2 \sum_{i,j} (A_{ij}B_{ij}) + 2q \sum_{i,j} B_{ij}^2$$

is a non-trivial linear function of q . The second order derivative is equal to $2 \sum_{i,j} B_{ij}^2$ and always takes positive values. Therefore, there exist exactly one local minimum of the original function.

In our approach, for a given CPT we try to identify the set of noisy-MAX parameters that minimizes the distances D_E or D_{KL} .

The problem amounts to finding the minimum of the distance which is a multidimensional function of the noisy-MAX parameters. We proved that for the Euclidean distance, there exists exactly one local minimum. Therefore, any mathematical optimization method ensuring convergence to a single minimum can be used. In case of KL divergence, we have no guarantee that there exists exactly one local minimum. But for the purpose of our experiments, this is a conservative assumption. To perform our experiments we implemented a simple gradient descent algorithm that takes a CPT as an input and produces noisy-MAX parameters and a measure of fit as an output.

4 HOW COMMON ARE NOISY-MAX GATES IN REAL MODELS

We decided to test several sizeable real world models in which probabilities were specified by an expert, learned from data, or combination of both. Three models that contained sufficiently large CPTs which were defined without parametric distributions were available to us: ALARM [12], HAILFINDER [11] and HEPAR II [8]. We verified by contacting the authors of these models that none of the CPTs in these networks were specified using the noisy-OR/MAX assumption. For each of the networks, we first identified all nodes that had at least two parents and then we applied our conversion algorithm to these nodes. HEPAR contains 31 such nodes, while ALARM and HAILFINDER contain 17 and 19 such nodes respectively. We tried to fit the noisy-MAX model to each of these nodes using both D_E and D_{KL} measures.

We used two criteria to measure the goodness of fit between a CPT and its MAX-based equivalent: (1) *Average*, the average Euclidean distance (with square root, not weighted by probabilities of parents instantiations) between the two corresponding parameters and (2) *Max*, the maximal absolute value of difference between two corresponding parameters, which is an indicator of the single worst parameter fit for a given CPT.

Fig. 2 shows the results for the three tested networks for Euclidean and KL measures respectively. The figures show the distance for all networks on one plot. The nodes in each of the networks are sorted according to the corresponding distance (*Average* or *MAX*) and the scale is converted to percentages. We can see for the *MAX* distance that for roughly 50% of the variables in two of the networks the greatest difference between two corresponding values in the compared CPTs was less than 0.1.

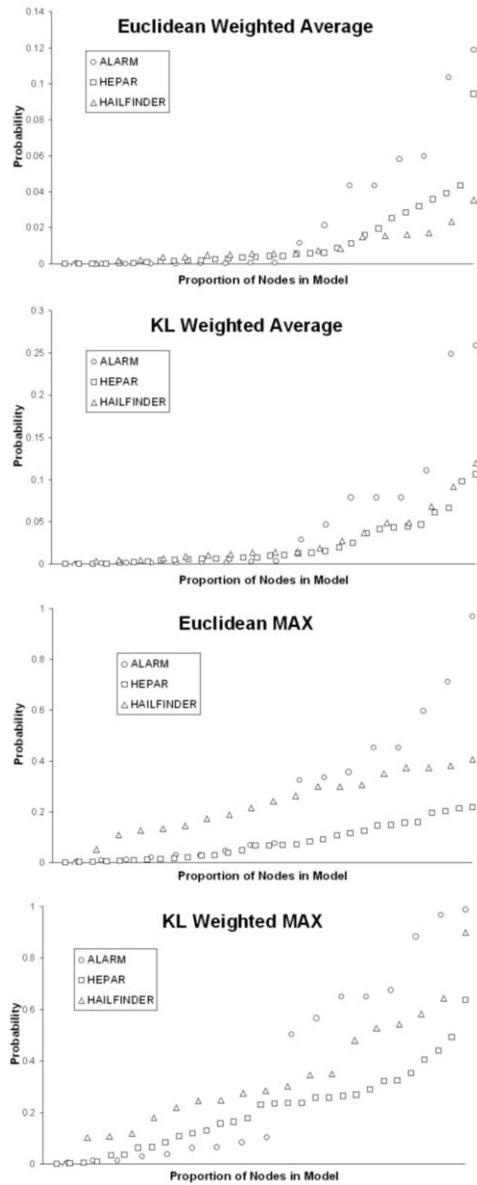


Figure 2. The Average and MAX distance for the nodes of the three analyzed networks obtained using weighted distances.

One possible explanation of our findings is that the noisy-MAX model is likely to fit well any randomly selected CPT. We decided to verify this by generating CPTs for binary nodes, with 2-5 parents (10,000 CPTs for every number of parents, for a total of 40,000 CPTs), whose parameters were sampled from the uniform distribution. Fig. 3 shows the results. On the X-axis there are generated CPTs sorted according to their fit to the noisy-OR using average and MAX measures. Except for the cases with two parents, the results are qualitatively different from the results we obtained using real-life models. They clearly indicate that approximating a randomly generated CPT by the noisy-OR is highly improbable. Additionally, these results can provide empirical grounds for interpretation of values of distance measures.

The small difference in the conditional probabilities does not necessarily imply that differences in the posterior probabilities will be of

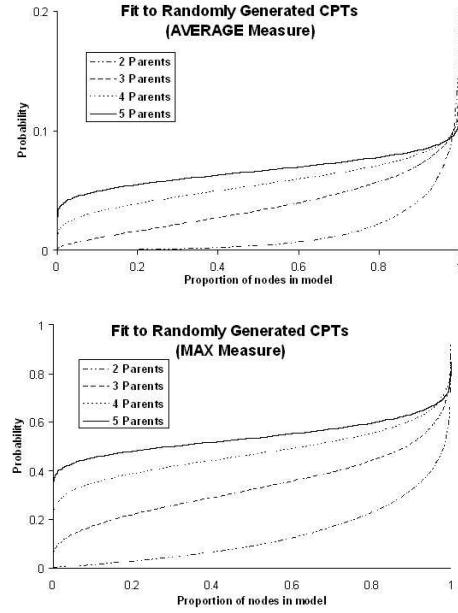


Figure 3. The Average and MAX distances for randomly generated CPTs.

a similar magnitude. We decided to test the accuracy of models with some nodes converted into the noisy-MAX. For each of the tested networks we converted one by one the selected nodes into noisy-MAX gates using Euclidean and KL measures. In this way, after each node had been converted, the new model was created. For each such model we generated random evidence using logic sampling for 10% of the nodes in the network and calculated the posterior probability distributions over the remaining nodes. We compared these posterior probabilities to those obtained in the original model, which we treated as the gold standard. We repeated the procedure described above 1000 times for each of the three models.

We decided to perform fit of the noisy-MAX model to CPTs without taking into account the probabilities of parent instantiations. One may argue that to answer the question if a local distribution fits the noisy-MAX should not take into account the proper distribution of parent variables. It is equivalent to assuming that the constants w_{P_i} in Eqs. 1 and 2 are always equal to 1. We decided to report these results together with the weighted distances. In the sequel, we will refer to uniformly weighted distributions as *simple* and the weighted according to probabilities of parent combinations as *weighted*.

Fig. 4 shows the results of tests for accuracy of posterior probabilities for the three networks. On the X-axis there are nodes sorted by the goodness of fit. On the Y-axis there is absolute average maximal error between posterior probabilities for 1,000 trials. We observe a consistent tendency that the accuracy of the posterior probabilities is poorer for nodes that have worse fit to the noisy-MAX. From these results one can conclude that the weighted KL divergence is superior to other distances, when it comes to CPTs which are good fits to the noisy-MAX (these at the left hand side of X-axis). The nodes on the right hand side are usually not of interest, because they represent nodes that are not good fit anyway. The nature of maximal error can explain the weighted distances' worse performance in case of HEPAR II network. The HEPAR II network has many small probabilities, and these lead to orders of magnitude differences in probabilities of parents' instantiations. The weighted distances work well on the

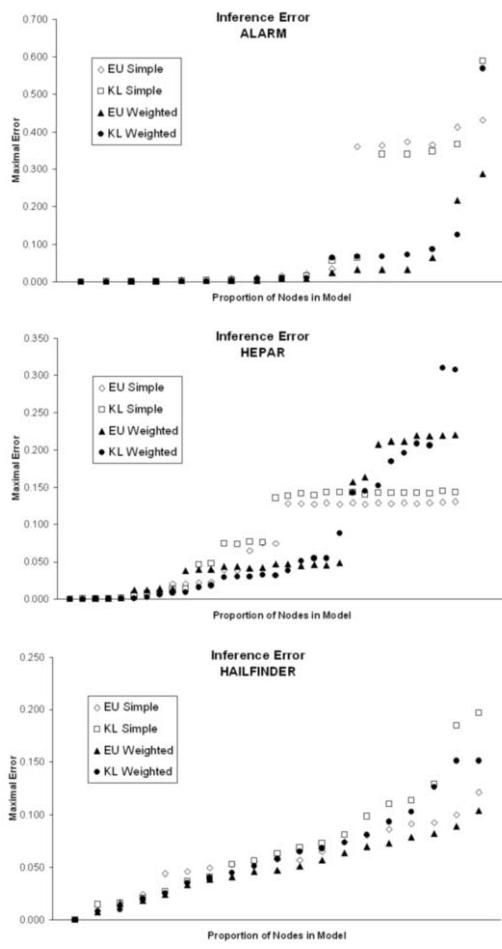


Figure 4. Accuracy of the posterior probabilities for the three networks. Evidence sampled from the posterior distribution.

average, but their performance can be worse for less likely scenarios and this additionally can be amplified by using the maximal measure, which captures the worst case scenario.

In the later version of the HEPAR II model domain expert's knowledge was used to identify variables that are candidates for the noisy-MAX distributions [8]. We had an access to this later version of the model and we could compare results obtained by our algorithm to the variables indicated by the expert. The surprising part is that the expert did not identify variables that were the best fit – she listed only 4 of the 10 variables with the best fit. Apparently the experts strategy was to convert as much as possible variables with large CPTs to the noisy-MAX, even though they were not a good fit. After some closer investigation, we concluded that these nodes are indeed a good fit to the noisy-MAX, but they required relatively complex manipulations on the order of states to fit the noisy-MAX model which was possibly beyond expert's modeling skills.

5 DISCUSSION

In this paper, we introduced two measures of distance between two CPTs – one based on the Euclidean distance and one based on the KL divergence. We proved that Euclidean distance between any CPT and a MAX-based CPT, as a function of the noisy-MAX parame-

ters of the latter, has exactly one minimum. We applied this result to an algorithm that given a CPT finds a noisy-MAX distribution that provides the best fit to it. Subsequently, we analyzed CPTs in three existing Bayesian network models using both measures. Our experimental results suggest that the noisy-MAX gate may provide a surprisingly good fit for as many as 50% of CPTs in practical networks. We demonstrated, that this result can not be observed in randomly generated CPTs. We tested the influence of accuracy of the approximation of a CPTs by noisy-MAX gates on the accuracy of posterior probabilities showing that models with some nodes converted to the noisy-MAX provide good approximation of the original models. Our results provide strong empirical support for the practical value of the noisy-MAX models. We showed that the relation defined by the noisy-MAX often approximates interactions in the modeled domain reasonably well. It seems to us, based on this result, that the independence of causal interactions is fairly common in real-world distributions.

ACKNOWLEDGEMENTS

This research was supported by the Air Force Office of Scientific Research under grant F49620-03-1-0187 and FA9550-06-1-0243.

REFERENCES

- [1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1988.
- [2] Y. Peng and J. A. Reggia, "Plausibility of diagnostic hypotheses," in *Proceedings of the 5th National Conference on AI (AAAI-86)*, Philadelphia, 1986, pp. 140–145.
- [3] M. Henrion, "Some practical issues in constructing belief networks," in *Uncertainty in Artificial Intelligence 3*, L. Kanal, T. Levitt, and J. Lemmer, Eds. New York, N. Y.: Elsevier Science Publishing Company, Inc., 1989, pp. 161–173.
- [4] F. J. Díez, "Parameter adjustment in Bayes networks. The generalized noisy OR-gate," in *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*. Washington D.C.: Morgan Kaufmann, San Mateo, CA, 1993, pp. 99–105.
- [5] F. J. Díez, J. Mira, E. Iturralde, and S. Zubillaga, "DIAVAL, a Bayesian expert system for echocardiography," *Artificial Intelligence in Medicine*, vol. 10, pp. 59–73, 1997.
- [6] M. Pradhan, G. Provan, B. Middleton, and M. Henrion, "Knowledge engineering for large belief networks," in *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*. San Francisco, CA: Morgan Kaufmann Publishers, 1994, pp. 484–490.
- [7] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper, "Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms," *Methods of Information in Medicine*, vol. 30, no. 4, pp. 241–255, 1991.
- [8] A. Onisko, M. J. Druzdzel, and H. Wasyluk, "Learning Bayesian network parameters from small data sets: Application of noisy-OR gates," *International Journal of Approximate Reasoning*, vol. 27, no. 2, pp. 165–182, 2001.
- [9] F. J. Díez and S. F. Galán, "Efficient computation for the noisy-MAX," *International Journal of Intelligent Systems*, vol. 18, no. 2, pp. 165–177, 2004.
- [10] D. Heckerman and J. S. Breese, "A new look at causal independence," in *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*. San Francisco, CA: Morgan Kaufmann Publishers, 1994, pp. 286–292.
- [11] I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper, "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks," in *Proceedings of the Second European Conference on Artificial Intelligence in Medical Care*, London, 1989, pp. 247–256.
- [12] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. Winkler, "Hailfinder: A Bayesian system for forecasting severe weather," in *International Journal of Forecasting*, Amsterdam, 1996, pp. 57–71.

5. Machine Learning

This page intentionally left blank

A Unified Model for Multilabel Classification and Ranking

Klaus Brinker¹ and Johannes Fürnkranz² and Eyke Hüllermeier¹

Abstract. Label ranking studies the problem of learning a mapping from instances to rankings over a predefined set of labels. Hitherto existing approaches to label ranking implicitly operate on an underlying (utility) scale which is not calibrated in the sense that it lacks a natural zero point. We propose a suitable extension of label ranking that incorporates the calibrated scenario and substantially extends the expressive power of these approaches. In particular, our extension suggests a conceptually novel technique for extending the common *learning by pairwise comparison* approach to the multilabel scenario, a setting previously not being amenable to the pairwise decomposition technique. We present empirical results in the area of text categorization and gene analysis, underscoring the merits of the calibrated model in comparison to state-of-the-art multilabel learning methods.

1 INTRODUCTION

Label ranking, a particular preference learning scenario [7], studies the problem of learning a mapping from instances (typically represented by feature vectors) to rankings over a finite number of predefined *labels*, in this context also referred to as *alternatives*. Approaches that operate in this framework include *Ranking by pairwise comparison* (RPC) as a natural extension of pairwise classification [6] and *constraint classification* which aims at learning a linear utility function for each label [8].

Although this framework is very general in the sense that simpler learning problems, such as classification or l -multilabel classification can be embedded as special cases, it is restricted to ranking the labels on a non-calibrated scale, that is, a scale which lacks a natural zero-point. Learning problems, such as conventional multilabel learning, which require a bipartite partition into complementary sets (relevant and non-relevant) do not admit a representation as special instances of either RPC or constraint classification. More generally, as a consequence of the underlying non-calibrated scales, both approaches cannot learn to determine the relevant/non-relevant cutoff, even though this information is specified in the training data.

We will present a general model avoiding the aforementioned drawbacks by incorporating a calibrated scale which contains a natural zero-point. Extending conventional label ranking approaches, this novel framework provides a means to represent and learn bipartite partitions of alternatives. In particular, it suggests a conceptually new technique for extending the common pairwise classification learning approach to the multilabel scenario, a setting previously not being amenable to a pairwise decomposition technique.

¹ Otto-von-Guericke-Universität Magdeburg, Germany, email: {brinker,huellerm}@iti.cs.uni-magdeburg.de

² TU Darmstadt, Germany, email: juffi@ke.informatik.tu-darmstadt.de

2 LABEL RANKING

In label ranking, the problem is to learn a mapping from instances $x \in \mathcal{X}$ to rankings \succ_x (total strict orders) over a finite set of labels $\mathcal{L} = \{\lambda_1, \dots, \lambda_c\}$, where $\lambda_i \succ_x \lambda_j$ means that, for instance x , label λ_i is preferred to λ_j . A ranking over \mathcal{L} can be represented by a permutation as there exists a unique permutation τ such that $\lambda_i \succ_x \lambda_j$ iff $\tau(\lambda_i) < \tau(\lambda_j)$, where $\tau(\lambda_i)$ denotes the position of the label λ_i in the ranking. The target space of all permutations over c labels will be referred to as \mathcal{S}_c .

It has been pointed out in several publications [8; 6; 3] that a variety of learning problems may be viewed as special cases of label ranking (perhaps supplemented by a straightforward projection of the output space \mathcal{S}_c) hence underscoring the importance of this setting. Among those are the following:

- *Multiclass Classification:* A single class label λ_i is assigned to each example x . This implicitly defines the set of preferences $R_x = \{\lambda_i \succ_x \lambda_j \mid 1 \leq j \neq i \leq c\}$. The output space \mathcal{S}_c is projected to the first component.
- *l-Multilabel Classification:* Each training example x is associated with a subset $P_x \subseteq \mathcal{L}$ of possible labels. This implicitly defines the set of preferences $R_x = \{\lambda_i \succ_x \lambda_j \mid \lambda_i \in P_x, \lambda_j \in \mathcal{L} \setminus P_x\}$. The output space is projected to the first l components.

RPC and constraint classification provide a general means to extend arbitrary (linear) binary classification algorithms to the label ranking scenario. Both approaches require (not necessarily complete) sets of pairwise preferences associated with the training instances to learn a ranking model which, as a post-processing step, may be projected from \mathcal{S}_c to the specific output space \mathcal{Y} .

The key idea of RPC is to learn, for each pair of labels (λ_i, λ_j) , a binary model $\mathcal{M}_{ij}(x)$ that predicts whether $\lambda_i \succ_x \lambda_j$ or $\lambda_j \succ_x \lambda_i$ for an input x . In order to rank the labels for a new instance, predictions for all pairwise label preferences are obtained and a ranking that is maximally consistent with these preferences is derived, typically by means of a voting scheme.³ This technique describes a natural extension of pairwise classification, i.e., the idea to approach a multiclass classification problem by learning a separate model for each pair of classes.

As we will see in the next section, conventional multilabel classification can not be embedded as a special case of the label ranking setting because, even though RPC or constraint classification could be used to rank the labels, they do not include a mechanism for extracting the set of relevant labels from this ranking.

³ To account for equal vote statistics, we consider random tie breaking in our implementation.

3 MULTILABEL CLASSIFICATION AND RANKING

Multilabel classification refers to the task of learning a mapping from an instance $x \in \mathcal{X}$ to a set $P_x \subset \mathcal{L}$, where $\mathcal{L} = \{\lambda_1, \dots, \lambda_c\}$ is a finite set of predefined labels, typically with a small to moderate number of alternatives. Thus, in contrast to multiclass learning, alternatives are not assumed to be mutually exclusive such that multiple labels may be associated with a single instance. The set of labels P_x are called *relevant* for the given instance, the set $N_x = \mathcal{L} \setminus P_x$ are the *irrelevant* labels.

A common approach to multilabel classification is *binary relevance learning* (BR). BR trains a separate binary relevance model \mathcal{M}_i for each possible label λ_i , using all examples x with $\lambda_i \in P_x$ as positive examples and all those with $\lambda_i \in N_x$ as negative examples. For classifying a new instance, all binary predictions are obtained and then the set of labels corresponding to positive relevance classification is associated with the instance. This scenario is, for example, commonly used for evaluating algorithms on the REUTERS text classification benchmark [10].

In the following, we will study the task of *multilabel ranking*, which is understood as learning a model that associates with a query input x both a ranking of the complete label set $\{\lambda_1, \dots, \lambda_c\}$ and a bipartite partition of this set into relevant and irrelevant labels. Thus, multilabel ranking can be considered as a generalization of both multilabel classification and ranking.

In conventional label ranking, a training example typically consists of an instance $x \in \mathcal{X}$, represented with a fixed set of features, and a set of pairwise preferences over labels $R_x \subset \mathcal{L}^2$, where $(\lambda, \lambda') \in R_x$ is interpreted as $\lambda \succ_x \lambda'$. In multilabel classification, the training information consists of a set P_x of relevant labels and, implicitly, a set $N_x = \mathcal{L} \setminus P_x$ of irrelevant labels. Note that this information can be automatically transformed into a set of preferences $\hat{R}_x = \{(\lambda, \lambda') \mid \lambda \in P_x \wedge \lambda' \in N_x\}$ (cf. Fig. 1 (a)).

While it is straightforward to represent the training information for multilabel classification as a preference learning problem, the algorithms that operate in this framework only produce a ranking of the available options. In order to convert the learned ranking into a multilabel prediction, the learner has to be able to autonomously determine a point at which the learned ranking is split into sets of relevant and irrelevant labels. Previous applications of ranking techniques to multilabel learning, such as [2], have ignored this problem and only focused on producing rankings, but not on determining this correct *zero point* for splitting the ranking.

Multilabel ranking can, for example, be realized if the binary classifiers provide real-valued confidence scores or *a posteriori* probability estimates for classification outcomes. Schapire & Singer [12] included an *ad hoc* extension to multilabel ranking in their experimental setup by ordering labels according to decreasing confidence scores.

In the following, we will introduce calibrated ranking, a conceptually new technique for extending the common pairwise learning approach to the multilabel scenario, a setting previously not being amenable to a pairwise decomposition approach. Within our framework, RPC can solve both multilabel classification and ranking problems in a consistent and generally applicable manner.

4 CALIBRATED LABEL RANKING

In this section, we will propose a general model avoiding the aforementioned drawbacks by incorporating a calibrated scale which con-

tains a natural zero-point. This zero-point provides a means to distinguish between the top and the complementary set of labels.

Let us proceed to a formal definition of the hypothesis space underlying the *calibrated label ranking framework*:

Definition 4.1 (Calibrated Label Ranking Model). Denote by \mathcal{X} a nonempty input space and by \mathcal{S}_c^0 the space of permutations over the set $\{\lambda_0, \lambda_1, \dots, \lambda_c\}$, that is, the original set of labels plus an additional virtual label λ_0 . Then, a model $h : \mathcal{X} \rightarrow \mathcal{S}_c^0$ is referred to as a calibrated label ranking model.

The key idea is to use the virtual label λ_0 as a split point between relevant and irrelevant labels: all relevant labels are preferred to λ_0 , which in turn is preferred to all irrelevant labels. Thus, a *calibrated ranking*

$$\lambda_{i_1} \succ \dots \succ \lambda_{i_j} \succ \lambda_0 \succ \lambda_{i_{j+1}} \succ \dots \succ \lambda_{i_c} \quad (1)$$

induces both a ranking among the labels,

$$\lambda_{i_1} \succ \dots \succ \lambda_{i_j} \succ \lambda_{i_{j+1}} \succ \dots \succ \lambda_{i_c}, \quad (2)$$

and a bipartite partition into

$$P = \{\lambda_{i_1}, \dots, \lambda_{i_j}\} \quad \text{and} \quad N = \{\lambda_{i_{j+1}}, \dots, \lambda_{i_c}\} \quad (3)$$

in a straightforward way.

As sketched in the previous section, the training information for a multilabel ranking problem consists of a set of preferences R_x , and subsets of labels $P_x, N_x \subset \mathcal{L}$ with $P_x \cap N_x = \emptyset$, which distinguish, respectively, *positive* labels that should be ranked above the zero-point element λ_0 and *negative* labels to be ranked below.⁴ The bipartite partitions associated with the training instances is used to, with the help of the virtual label λ_0 , induce additional constraints: the calibrated classifier h should predict $\lambda \succ_x \lambda_0$ for all $\lambda \in P_x$ and vice-versa $\lambda_0 \succ_x \lambda'$ for all $\lambda' \in N_x$ (cf. Fig. 1 (b)). Moreover, as a consequence of transitivity, it should predict $\lambda \succ_x \lambda'$ for all $\lambda \in P_x$ and $\lambda' \in N_x$ (Fig. 1 (c)). Combining the new partition-induced preference constraints with the original set of pairwise preferences for the training data, i.e.,

$$\begin{aligned} R'_x &\stackrel{\text{def}}{=} R_x \cup \{(\lambda, \lambda_0) \mid \lambda \in P_x\} \\ &\quad \cup \{(\lambda_0, \lambda') \mid \lambda' \in N_x\} \\ &\quad \cup \{(\lambda, \lambda') \mid \lambda \in P_x \wedge \lambda' \in N_x\}, \end{aligned} \quad (4)$$

the calibrated ranking model becomes amenable to previous approaches to the original label ranking setting: The calibrated ranking model can be learned by solving a conventional ranking problem in the augmented calibrated hypothesis space, which may be viewed as a ranking problem with $c + 1$ alternatives, with respect to the modified sets of constraints R'_x on the original labels $\lambda_1, \dots, \lambda_c$ and the virtual label λ_0 . Therefore, this unified approach to the calibrated setting enables many existing techniques, such as RPC and constraint classification [1], to incorporate and exploit partition-related preference information and to generalize to settings where predicting the zero-point is required. We will discuss an exemplary application of this framework to pairwise ranking in Section 5.

⁴ In general, we do not need to assume complete training data, neither for the sets of preferences (R_x might even be empty) nor for the partitions (which do not necessarily have to cover all the labels, i.e., $P_x \cup N_x \neq \mathcal{L}$). Besides, in a *noisy learning scenario*, it may happen that $(\lambda', \lambda) \in R_x$ even though $\lambda \in P_x$ and $\lambda' \in N_x$. In this paper, we will not further consider these cases, and assume a strict multilabel scenario.

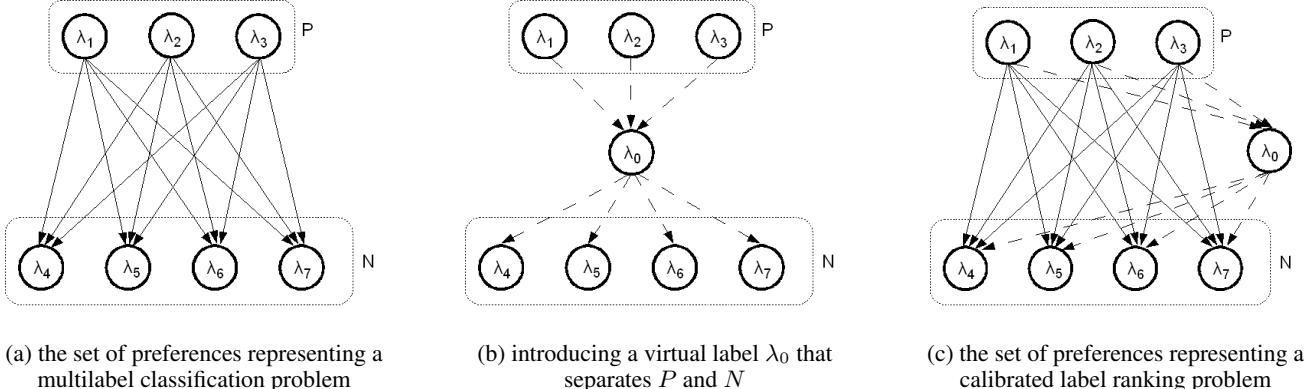


Figure 1. Calibrated Label Ranking

4.1 Relation to Binary Relevance Learning

Conventional pairwise label ranking learns a binary preference model M_{ij} for all combinations of labels λ_i and λ_j with $1 \leq i < j \leq c$,⁵ where instances x with $(\lambda_i, \lambda_j) \in R_x$ are associated with positive and those with $(\lambda_j, \lambda_i) \in R_x$ with negative class labels (cf. Figure 1 (b)). In the calibrated scenario, the partition-related constraints with respect to λ_0 as defined in (4) are required to learn an additional set of binary preference models M_{0j} with $1 \leq j \leq c$. It is important to notice, that these additional models are identical to the common binary-relevance models M_j .

Theorem 4.2. *The models M_{0j} that are learned by a pairwise approach to calibrated ranking, and the models M_j that are learned by conventional binary ranking are equivalent.*

Proof. Each training example x , for which label λ_j is relevant, is, by definition, a positive example in the training set for model M_j . The calibrated ranking approach adds the preference $\lambda_j \succ_x \lambda_0$, which means that x will be a negative example for M_{0j} . Similarly, if λ_j is irrelevant, x is negative for M_j and positive for M_{0j} . Assuming a symmetric learner, the learned models will be the equivalent in the sense that $M_j = -M_{0j}$. \square

Thus, calibrated RPC may be viewed as a method for combining RPC with conventional binary ranking, in the sense that the binary models that are learned for RPC, and the binary models that are learned for BR, are pooled into a single ensemble. However, CRPC provides a new interpretation to the BR models, that not only allows for ranking the labels, but also to determine a suitable split into relevant and irrelevant categories.

By training a larger number of pairwise models, the calibrated extension of RPC achieves two potential advantages over simple relevance learning. Firstly, it provides additional information about the ranking of labels. Secondly, it may also improve the discrimination between relevant and irrelevant labels. In fact, it is legitimate to assume (and indeed supported by empirical evidence in the next section) that the additional binary models can somehow “stabilize” the related classification. For example, while an error of model $M_j = -M_{0j}$ definitely causes a misclassification of label λ_j in simple relevance learning, this error might be compensated by the models M_{ij} , $1 \leq i \neq j \leq c$, in the case of RPC. The prize to pay is, of course, a higher computational complexity, which, as we will show in the next section, depends on the maximum number of labels for an example.

⁵ The case $i > j$ is typically not required as a consequence of the symmetry of binary classifiers with respect to positive and negative instances.

4.2 Computational Complexity

In this section, we will derive bounds for the number of training examples that are constructed for training the pairwise classifiers. The total computational complexity depends on the complexity of the base classifier used for processing these examples. In brief, super-linear classifiers like SVMs will profit from distributing the workload on many small problems instead of fewer larger problems as for the BR approach.

Theorem 4.3. *CRPC is trained on $O(lcn)$ examples, where $l = \max_x |P_x|$ is the maximum number of relevant labels that may be associated with a single training example, c is the number of possible labels, and n is the number of original training examples.*

Proof. In previous work, it has been shown that training a pairwise classifier requires $O(cn)$ training examples [5]. To see this, note that each of the n original training examples will only appear in the training sets of $c - 1$ models, therefore the total number of training examples that have to be processed is $(c - 1)n$.

For multilabel classification, RPC will compare a training example's $|P_x|$ relevant labels to all $|N_x| = c - |P_x|$ labels that are not relevant for this example. In addition, CRPC will include every example in the c training sets of the models M_{0j} , $j = 1 \dots c$.

Thus, each example occurs in $|P_x| \times |N_x| + c = |P_x|(c - |P_x|) + c < |P_x|c + c < (l + 1)c$ training sets, and the total number of training examples is therefore $O(lcn)$. \square

This result shows that the complexity of training CRPC depends crucially on the maximum number of labels in the training examples. For the case of $l = 1$, i.e., for conventional pairwise classification, we get the linear bound that was shown in [5].⁶ For multilabel classification with a maximum number of l labels per example, we still have a bound that is linear in the number of examples and the number of classes, i.e., the complexity is within a factor of l of the $O(cn)$ examples needed for training a BR. We would like to point out that in many practical applications, the bound l is determined by the procedure that is used for labeling the training examples, and is independent of c . A typical example is the number of keywords that are assigned to a text, which is rarely ever more than ten.

Thus, for practical purposes, the complexity of CRPC is within a constant factor of l of the complexity of BR. Of course, the worst-case complexity, which would occur when all possible label subsets are *a priori* equally likely, is $O(c^2n)$.

⁶ Note that the O -notation hides, in this case, a constant factor of two, which results from the introduction of the virtual label λ_0 .

Table 1. Experimental Results on the Yeast Dataset.

	Degree	1	2	3	4	5	6	7	8	9	Optimum
CRPC	PREC	0.762*	0.768*	0.760*	0.767*	0.772*	0.772*	0.773	0.770	0.767	0.773
	RANKLOSS	0.168*	0.164*	0.170*	0.164*	0.159*	0.158*	0.158*	0.160	0.162	0.158
	ONEERROR	0.230	0.224*	0.230*	0.229*	0.229*	0.233	0.230	0.234	0.234	0.224
	HAMLOSS	0.199	0.197	0.208*	0.204*	0.199*	0.194	0.193	0.192	0.193	0.192
BR	PREC	0.746	0.755	0.733	0.742	0.755	0.762	0.768	0.772	0.771	0.772
	RANKLOSS	0.199	0.183	0.197	0.188	0.178	0.171	0.165	0.161	0.160	0.160
	ONEERROR	0.241	0.248	0.277	0.268	0.244	0.236	0.229	0.227	0.229	0.227
	HAMLOSS	0.199	0.198	0.219	0.209	0.202	0.196	0.192	0.192	0.192*	0.192

Bold face indicates superior performance comparing models with the same kernel parameters (except for the optimum-related column).

Stars indicate statistical significance at the $\alpha = 0.05$ level using a paired t-test.

Table 2. Experimental Results on the Reuters2000 Dataset.

	C	1	10	100	Optimum
CRPC	PREC	0.943	0.944	0.943	0.944
	RANKLOSS	0.031	0.031	0.031	0.031
	ONEERROR	0.052	0.052	0.052	0.052
	HAMLOSS	0.067	0.069	0.070	0.067
BR	PREC	0.940	0.935	0.933	0.940
	RANKLOSS	0.035	0.038	0.039	0.035
	ONEERROR	0.053	0.061	0.063	0.053
	HAMLOSS	0.067	0.069	0.071	0.067

Note: Bold face indicates superior performance comparing models with the same kernel parameters (except for the optimum-related column).

5 EMPIRICAL EVALUATION

The purpose of the following section is to provide an empirical comparison of calibrated RPC with the common binary-relevance approach in the domain of multilabel classification and ranking. The datasets that were included in the experimental setup cover two application areas in which multilabeled data are frequently observed: *text categorization* and *bioinformatics*.

- **Yeast:** The Yeast gene functional multiclass classification problem consists of 1500 genes in the training and 917 in the test set, represented by 103 features, where each gene is associated with a subset of the 14 functional classes considered [4].
- **Reuters2000:** The Reuters Corpus Volume I is one of the currently most widely used test collection for text categorization research. As the full corpus contains more than 800000 newswire documents, we restricted our experiments to a subset distribution of five times 3000 training and 3000 test documents which is publicly available as a preprocessed version.⁷ The documents are represented using stemmed word frequency vectors (normalized to unit length) with a TFIDF weighting scheme and elimination of stopwords resulting in 47152 features. Each document is associated with a subset of the 101 categories present in the dataset. The number of categories was reduced to 10 in our experiments by sequentially grouping the original categories into buckets of roughly equal size where each bucket corresponds to a single new label which is associated with positive relevance if at least one of the labels in the bucket is relevant.

We replicated the experimental setup in [4] to conduct the empirical evaluation on the Yeast dataset where a predefined split into training and test data was given. The four different measures considered cover both multilabel classification and ranking performance:

Preliminaries: For a multilabel ranking model h and a given instance x , let $\tau(\lambda_i)$ denote the position of λ_i in the predicted ranking (with

λ_0 being removed from the ranking), $\tau^{-1}(i)$ the label λ having assigned position i , and P the set of relevant labels as predicted by CRPC.

Precision (PREC) assesses the multilabel ranking performance and is a frequently used measure in Information Retrieval:

$$\text{PREC}(h, x, P_x) \stackrel{\text{def}}{=} \frac{1}{|P_x|} \sum_{\lambda \in P_x} \frac{|\{\lambda' \in P_x | \tau(\lambda') \leq \tau(\lambda)\}|}{\tau(\lambda)} \quad (5)$$

The ranking loss (RANKLOSS) also measures ranking performance as it evaluates the average fraction of pairs of labels which are not correctly ordered:

$$\text{RANKLOSS}(h, x, P_x) \stackrel{\text{def}}{=} \frac{|\{(\lambda, \lambda') \in P_x \times N_x | \tau(\lambda) > \tau(\lambda')\}|}{|P_x||N_x|}$$

The one-error loss (ONEERROR) evaluates the multilabel ranking performance from a restricted perspective as it only determines if the top-ranked label is actually relevant:

$$\text{ONEERROR}(h, x, P_x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \tau^{-1}(1) \notin P_x, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

The Hamming loss (HAMLOSS) assesses the multilabel classification performance in terms of the average binary (non-)relevant error:

$$\text{HAMLOSS}(h, x, P_x) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{L}|} |P \Delta P_x| \quad (7)$$

In compliance with [4], support vector machines (SVMs) [13] provided the underlying classifiers for the binary-relevance multilabel model, as well as for the calibrated ranking model which was built on top of SVMs with polynomial kernels. The degree of the kernel varied from 1 to 9 and the margin-error penalty was fixed to $C = 1$. Linear kernels with $C \in \{1, 10, 100\}$ were used on the Reuters2000 dataset as they have demonstrated excellent performance in many publications studying text mining problems [9].

The empirical results on the Yeast dataset, depicted in Table 1, demonstrate that the calibrated ranking model is a promising alternative to the common binary-relevance approach to multilabel classification and ranking. Over a wide range of parameter values, the performance of the calibrated ranking is superior to the binary-relevance approach and comparatively stable, thus indicating a high level of robustness. For the optimal choice of parameters with respect to the testset (which could be approximately determined by cross-validation or leave-one-out estimation in practice), the gap is decreasing while calibrated ranking still outperforms its binary-relevance counterpart on three of the four evaluation measures. Interestingly, the only measure for which both approaches achieve comparable accuracy, namely the Hamming loss, evaluates the multilabel accuracy

⁷ <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

in a manner that seems to be more tailored to the binary-relevance approach from a theoretical point of view as it computes the average (independently measured) binary-relevance error. Note that although the pattern observed in Table 1 might suggest that further increasing the degree of the polynomial kernel could improve the experimental results of the binary-relevance approach, this assumption could not be validated in an additional experiment. Despite that we could not set up a perfect replication of [4] as certain parameters (e.g., the margin-penalty values C) were not published, we nevertheless note that the performance of calibrated RPC compares favorably to those reported in [4] for the ranking and Hamming loss, while the results in terms of precision are comparable to those of the herein proposed SVM ranking algorithm. With respect to the one-error loss the performance is slightly worse. Moreover, there is a substantial margin between the performance of Boostexter [12], a boosting-based multilabel learning system, on the Yeast benchmark dataset (as published in [4]) and the calibrated ranking technique with respect to all four evaluation measures considered in the experimental setup.

The empirical results on the Reuters2000 dataset (see Table 2) clearly support the conclusions drawn from the Yeast dataset. In terms of the Hamming loss, both approaches achieve comparable accuracy, while calibrated ranking outperforms binary-relevance multilabel learning for the remaining evaluation measures. Again, the margin is typically smaller for the optimal choice of parameter values, whereas the difference in accuracy is larger when comparing the results with the parameter C being fixed. Moreover, calibrated ranking tends to be more stable than binary-relevance learning with respect to the choice of the underlying binary classifiers. Furthermore, additional experiments using different methods for reducing the number of labels, such as selecting only the most frequent relevant labels (cf., [12]), showed qualitatively comparable outcomes.

6 RELATED WORK

Schapire & Singer [11] derived a family of multilabel extensions in the framework of AdaBoost (referred to as AdaBoost.MH and AdaBoost.MR) which provided the algorithmic basis for the Boostexter text and speech categorization system [12]. For the online learning setting, a computationally efficient class of perceptron-style algorithms for multilabel classification and ranking was proposed in [2]. In a manner similar to the above mentioned approaches, the multilabel generalization of support vector machines advocated by Elisseeff & Weston [4] exploits comparative preferences among pairs of labels as defined for the multilabel case in Section 2, while lacking a natural approach to determine the relevance zero-point in the multilabel ranking.

7 CONCLUDING REMARKS

We have proposed a unified extension to overcome the severe restriction on the expressive power of previous approaches to label ranking induced by the lack of a calibrated scale. This unified approach to the calibrated ranking setting enables general ranking techniques, such as ranking by pairwise comparison and constraint classification, to incorporate and exploit partition-related preference information and to generalize to settings where predicting the zero-point is required. In particular, the calibrated extension suggests a conceptually novel technique for extending the common learning by pairwise comparison technique to the multilabel scenario, a setting previously not being amenable to pairwise decomposition. A particular limitation of the binary-relevance extension to multilabel ranking, which is not

shared by the calibrated framework, consists in the fact that it only applies to soft-classifiers providing confidence scores in the prediction. Exploring the benefits of calibrated ranking with binary-valued classifiers is a promising aspect of future work. Experimental results in the areas of text categorization and gene analysis underscore the merits of our calibrated framework from an empirical point of view.

ACKNOWLEDGEMENTS

This research was supported by the German Research Foundation (DFG) and Siemens Corporate Research (Princeton, USA).

REFERENCES

- [1] Klaus Brinker and Eyke Hüllermeier, ‘Calibrated Label-Ranking’, *Proceedings of the NIPS-2005 Workshop on Learning to Rank*, S. Agarwal and C. Cortes and R. Herbrich (eds.), pp. 1–6, Whistler, BC, Canada, (2005).
- [2] Koby Crammer and Yoram Singer, ‘A family of additive online algorithms for category ranking’, *Journal of Machine Learning Research*, **3**, 1025–1058, (2003).
- [3] Ofer Dekel, Christopher Manning, and Yoram Singer, ‘Log-linear models for label ranking’, in *Advances in Neural Information Processing Systems 16*, eds., Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, MIT Press, Cambridge, MA, (2004).
- [4] André Elisseeff and Jason Weston, ‘A kernel method for multi-labelled classification’, in *Advances in Neural Information Processing Systems 14*, eds., T. G. Dietterich, S. Becker, and Z. Ghahramani, pp. 681–687, Cambridge, MA, (2002). MIT Press.
- [5] Johannes Fürnkranz, ‘Round robin classification’, *Journal of Machine Learning Research*, **2**, 721–747, (2002).
- [6] Johannes Fürnkranz and Eyke Hüllermeier, ‘Pairwise preference learning and ranking’, in *Proceedings of the 14th European Conference on Machine Learning (ECML 2003)*, pp. 145–156, Cavtat, Croatia, (2003). Springer-Verlag.
- [7] Johannes Fürnkranz and Eyke Hüllermeier, ‘Preference learning’, *Künstliche Intelligenz*, **19**(1), 60–61, (2005).
- [8] Sariel Har-Peled, Dan Roth, and Dav Zimak, ‘Constraint classification: A new approach to multiclass classification and ranking’, in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, (2002).
- [9] Thorsten Joachims, ‘Text categorization with support vector machines: Learning with many relevant features’, in *Proceedings of the European Conference on Machine Learning (ECML 1998)*, eds., Claire Nédellec and Céline Rouveiro, pp. 137–142, Berlin, (1998). Springer.
- [10] David D. Lewis. Reuters-21578 text categorization test collection. README file (V 1.2), available from <http://www.research.att.com/~lewis/reuters21578/README.txt>, September 1997.
- [11] Robert E. Schapire and Yoram Singer, ‘Improved boosting using confidence-rated predictions’, *Machine Learning*, **37**(3), 297–336, (1999).
- [12] Robert E. Schapire and Yoram Singer, ‘BoosTexer: A boosting-based system for text categorization’, *Machine Learning*, **39**(2/3), 135–168, (2000).
- [13] Bernhard Schölkopf and Alexander J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002.

Learning by Automatic Option Discovery from Conditionally Terminating Sequences

Sertan Girgin¹ and Faruk Polat² and Reda Alhajj³

Abstract. This paper proposes a novel approach to discover options in the form of conditionally terminating sequences, and shows how they can be integrated into reinforcement learning framework to improve the learning performance. The method utilizes stored histories of possible optimal policies and constructs a specialized tree structure online in order to identify action sequences which are used frequently together with states that are visited during the execution of such sequences. The tree is then used to implicitly run corresponding options. Effectiveness of the method is demonstrated empirically.

1 Introduction

Reinforcement learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment by gaining percepts and rewards from the world and taking actions to affect it [3, 9]. In most of the realistic and complex domains, the task that the agent is trying to solve is composed of various subtasks and has a hierarchical structure formed by the relations between them [1]. Each of these subtasks repeats many times at different regions of the state space. Although, all instances of the same subtask, or similar subtasks, have almost identical solutions (sub-behaviors), without any (self) guidance an agent has to learn these solutions independently, going through similar learning stages again and again. This situation affects the learning process in a negative way, making it difficult to converge to optimal behavior in a reasonable time.

The main reason of the problem is the lack of abstractions, that would allow to share solutions between similar subtasks. *Temporally abstract actions* are macro actions that generalize primitive actions and last for a period of time [10]. In most applications, they are part of the problem specification and provided by the system developer prior to learning based on extensive domain knowledge; this becomes more difficult as the complexity of the problem increases. An alternative way is to construct options automatically using domain information acquired during learning. Most of the research conducted on this topic has focused on two main approaches. The first approach starts by identifying possible subgoals of the problem, then abstractions solving them are generated and utilized. Various statistics (such as visit frequencies of states) [5, 8] or graph theoretical notions (such as bottleneck states connecting the strongly connected components of the state space) [6, 7] are used to identify subgoals. Solutions of the subgoals are explicitly generated by means of additional rein-

forcement learning processes, such as action replay executed on restricted sub-problems with artificial rewards. Although these methods are effective in general, subtasks with similar policies but different subgoals result in independent abstractions. In relatively less explored second approach, temporal abstractions are generated directly, without identifying subgoals, by analyzing common parts of multiple policies. An example of this approach is proposed by McGovern, where sequences that occur frequently on successful action trajectories are detected, and options are created for those which pass a static filter that eliminates sequences leading to similar results [4]. One drawback of this method is that common action sequences are identified at regular intervals, which is a costly operation and requires all state and action histories be stored starting from the beginning of learning. Also, since every prefix of a frequent sequence has at least the same frequency, the number of possible options increases rapidly unless limited in a problem specific way.

In this work, we propose a method which efficiently discovers useful options in the form of conditionally terminating action sequences without storing all observations. From the histories of states, actions and reward, we first generate trajectories of possible optimal policies, and then convert them into a tree structure. This helps to identify and compactly represent frequently used sub-sequences of actions together with states that are visited during their execution. As learning progresses, this tree is constantly updated and used to implicitly run represented options. We demonstrate the effectiveness of this approach by reporting test results on two domains.

The rest of the paper is organized as follows: In Section 2 we describe the standard reinforcement learning framework of discrete time, finite Markov decision processes. The notion of options, and specifically conditionally terminating sequences, is defined in Section 3. A method to discover the options online during the learning is described in Section 4. We present experimental results in Section 5. Section 6 is conclusions.

2 Background

A *Markov decision process*, MDP, is a tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is a state transition function such that $\forall s \in S, \forall a \in A, \sum_{s' \in S} T(s, a, s') = 1$, and $R : S \times A \rightarrow \mathbb{R}$ is a reward function. $T(s, a, s')$ denotes the probability of making a transition from state s to state s' using action a . $R(s, a)$ is the *immediate* reward received when action a is executed in state s . A (stationary) *policy*, $\pi : S \times A \rightarrow [0, 1]$, is a mapping that defines the probability of selecting an action from a particular state. If $\forall s \in S, \pi(s, a_s) = 1$ and $\forall a \in A, a \neq a_s, \pi(s, a) = 0$ then π is called a *deterministic policy*. The *value* of a policy π at state

¹ Middle East Technical University, Ankara, Turkey; and University of Calgary, Calgary, Alberta, Canada, girgins@cpsc.ucalgary.ca

² Middle East Technical University, Ankara, Turkey, polat@ceng.metu.edu.tr

³ University of Calgary, Calgary, Alberta, Canada; and Global University, Beirut, Lebanon, alhajj@cpsc.ucalgary.ca

s , denoted $V^\pi(s)$, is the expected infinite discounted sum of reward that the agent will gain if it starts in state s and follows π [3]. It is defined as: $V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t E(r_t | \pi, s_0 = s)$, where r_t is the reward received at time t , and γ is the discount factor. Let $Q^\pi(s, a)$ denote the expected infinite discounted sum of reward that the agent will gain if it selects action a at s , and then follows π : $Q^\pi(s, a) = (R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\pi(s'))$; then, we have $V^\pi(s) = \sum_{a \in A} \pi(s, a) Q^\pi(s, a)$. $V^\pi(s)$ and $Q^\pi(s, a)$ are called policy's state value function and action value function, respectively. In a Markov decision process, the objective of an agent is to find an *optimal policy*, π^* , which maximizes state value function for all states (i.e., $\forall \pi, \forall s \in S, V^{\pi^*}(s) \geq V^\pi(s)$). Every MDP has a deterministic stationary optimal policy; and for all optimal policies, the state value function is identical; and $\forall s \in S$, the following Bellman equation holds: $V^*(s) = \max_{a \in A} (R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s')) = \max_{a \in A} Q^*(s, a)$. V^* and Q^* are called the optimal value functions. Using Q^* , π^* can be specified as $\pi^*(s) = \arg \max_{a \in A} Q^*(s, a)$. When the reward function R and the state transition function T are known, π^* can be found by using *dynamic programming* techniques [3]. When such information is not readily available, Monte Carlo or temporal-difference (TD) learning methods are used. Instead of requiring complete knowledge of the underlying model, these approaches rely on experience in the form of sample sequences of states, actions, and rewards collected from on-line or simulated trial-and-error interactions with the environment.

3 Options

The *options* framework of Sutton et. al. [10] extends the theory of reinforcement learning to include temporally extended courses of action. Formally, an option is a tuple $\langle I, \pi, \beta \rangle$, where $I \subseteq S$ is the set of states that the option can be initiated, called *initiation set*, π is the option's local policy and β is the termination condition. Once an option is initiated by an agent at a state $s \in I$, π is followed and actions are selected according to π until the option terminates (stochastically) at a specific condition determined by β . In a *Markov option*, action selection and option termination decisions are made solely on the basis of the current state, i.e., $\pi : S \times A \rightarrow [0, 1]$, and $\beta : S \rightarrow [0, 1]$. During option execution, if the environment makes a transition to state s , then the Markov option terminates with probability $\beta(s)$ or else continues, determining the next action a with probability $\pi(s, a)$.

Markov options are limited in their ability to represent some useful abstractions, such as terminating after a given number of steps, or carrying out a sequence of actions irrespective of the consequences (as in open-loop control). For more flexibility, *Semi-Markov options* extend Markov policies and allow π and/or β to depend on all prior events since the option was initiated. Let $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \dots, r_\tau, s_\tau$ be the sequence of states, actions and rewards observed by the agent starting from time t until τ . As in [10], we will call this sequence the *history* from t to τ , and denote it by $h_{t\tau}$. If the set of all possible histories is denoted by Ω , then in a Semi-Markov option β is defined over Ω as, $\beta : \Omega \rightarrow [0, 1]$, and similarly π is a function of Ω , $\pi : \Omega \times A \rightarrow [0, 1]$.

In this work, we concentrate on Semi-Markov options in the form of conditionally terminating sequences. Following [4], we define a *conditionally terminating sequence*, *ct-sequence* in short, of length n as a sequence of initiation set and action tuples $\sigma = \langle I_1, a_1 \rangle \langle I_2, a_2 \rangle \dots \langle I_n, a_n \rangle$, where $I_i \subseteq S$ and $a_i \in A$. At step i of the execution of σ , if current state s is in I_i then agent selects action a_i and the sequence advances to the next step; otherwise the

sequence terminates. Here, $a_1 a_2 \dots a_n$ is called the *action sequence* of σ and denoted by A_σ . $I_\sigma = I_1$ and $a_\sigma = a_1$ are the initiation set and first action of σ , respectively. Let $I_{\sigma,i}$ and $a_{\sigma,i}$ denote the i^{th} initiation set and action of σ . For every ct-sequence σ one can define a corresponding Semi-Markov option $\sigma_\sigma = \langle I, \pi, \beta \rangle$ as follows: $I = I_{\sigma,1}$,

$$\begin{aligned} \pi(h_{t\tau}, a) &= \begin{cases} 1, & \text{if } \beta(h_{t\tau}) = 1 \wedge a = a_{\sigma, \tau-t+1} \\ 0, & \text{otherwise} \end{cases} \\ \beta(h_{t\tau}) &= \begin{cases} 1, & \text{if } \tau - t < n \wedge s_\tau \in I_{\sigma, \tau-t+1} \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Conditionally terminating sequences can be used to represent frequently occurring and useful patterns of actions in a reinforcement learning problem. For example, consider a 5×5 grid world, in which the agent's goal is to reach the top rightmost square as soon as possible (i.e., with minimum number of actions). At each time step, it can move to one of the four neighboring cells if there is a cell on that direction. Assume the set of states consists of pairs representing the row and column indexes of the cells on the grid: $S = \{(0, 0), \dots, (4, 4)\}$; $A = \{n, s, e, w\}$ is the set of possible actions, and $[(r_1, c_1), (r_2, c_2)]$ denotes the rectangular region on the grid with corners at (r_1, c_1) and (r_2, c_2) . In order to reach the goal cell, one of the useful action patterns that can be used by the agent is to move diagonally in the north-east direction, i.e., e followed by n or vice versa. These patterns can be represented by ct-sequences: $\sigma_{en} = \langle [(0, 0), (3, 3)], e \rangle \langle [(0, 1), (3, 4)], n \rangle$ and $\sigma_{ne} = \langle [(0, 0), (3, 3)], n \rangle \langle [(1, 0), (4, 3)], e \rangle$, respectively.

Let $|\sigma|$ be the length of a ct-sequence σ , and $\sigma^{i,j} = \langle I_{\sigma,i}, a_{\sigma,i} \rangle \dots \langle I_{\sigma,j}, a_{\sigma,j} \rangle$ denotes the ct-sequence obtained from σ by taking initiation set and action tuples starting from i up to j . We use σ^i as a short notation for $\sigma^{i,|\sigma|}$. Given two ct-sequences u and v , if $|u| \leq |v|$ and the action sequence of v starts with the action sequence of u , then u is called a *predecessor* of v , indicated by $u \preceq v$. For example, moving diagonally in the north-east direction two times can be represented by a ct-sequence $\sigma_{enen} = \langle [(0, 0), (2, 2)], e \rangle \langle [(0, 1), (2, 3)], n \rangle \langle [(1, 1), (3, 3)], e \rangle \langle [(1, 2), (3, 4)], n \rangle$, and $\sigma_{en} \preceq \sigma_{enen}$. Although not necessary, initiation sets of a preceded sequence generally are supersets of corresponding initiation sets of the successor, as in case of σ_{en} and σ_{enen} .

If a sequence is a predecessor of another one, then it is possible to merge the two sequences to obtain a new ct-sequence that reflects behavior of both sequences. Let u and v be two ct-sequences such that $u \preceq v$. Their union is defined as $u \cup v = \langle I_{u,1} \cup I_{v,1}, a_{u,1} \rangle \dots \langle I_{u,|u|} \cup I_{v,|u|}, a_{u,|u|} \rangle \langle I_{v,|u|+1}, a_{v,|u|+1} \rangle \dots \langle I_{v,|v|}, a_{v,|v|} \rangle$. When started at a state s , $u \cup v$ behaves like u if $s \in I_u \setminus I_v$, and like v otherwise. Note $u \cup v$ may behave like v even if $s \notin I_v$, under the condition that after executing $|u|$ steps, v can continue thereafter. Also, depending on the actual observed states, the course of action followed by $u \cup v$ may switch between u and v . In particular, let $s_{t+1} s_{t+2} \dots s_{t+k}$ be the sequence of states observed during the execution of $u \cup v$, and suppose that there exist states s_{t+i} and s_{t+j} such that $s_{t+i} \in I_{u,i} \setminus I_{v,i}$ and $s_{t+j} \in I_{v,j} \setminus I_{u,j}$, where $1 \leq j < i < k$. Then, if started at s_t , v would terminate at s_{t+i} , and similarly u would terminate at s_{t+j} without having opportunity to advance to step k . Note that $u \cup v$ switches to v at step i and to u at step j . Therefore, the union of two ct-sequences not only combines but also generalizes their behavior in favor of longer execution patterns, when possible.

Now suppose that we are given a set of ct-sequences $\Sigma = \{\sigma_1, \dots, \sigma_n\}$ to be used in a RL problem. Furthermore, assume that the option to be executed at state s is chosen on the basis of

the probability distribution $P(s, \cdot)$ determined by a given function $P : S \times \Sigma \rightarrow [0, 1]$. The set Σ can be represented in a compact form using a *stochastic ct-sequence* which extends conditionally terminating sequences to allow different action sequences be followed depending on the history of events back to starting of its execution. Let \bar{I}_ς denote the set of states at which a stochastic ct-sequence ς can be initiated, and \bar{f}_ς be the set of possible first actions that can be selected by ς . A stochastic ct-sequence is defined inductively as:

1. A ct-sequence σ is a stochastic ct-sequence; its initiation set and first action set are I_σ and $\{f_\sigma\}$, respectively.
2. Given a ct-sequence u and a stochastic ct-sequence v , their concatenation $u \circ v$, defined as executing u followed by v , is a stochastic ct-sequence with $\bar{I}_{u \circ v} = \bar{I}_u$ and $\bar{f}_{u \circ v} = \bar{f}_u$.
3. For a given set of stochastic ct-sequences $\Xi = \{\varsigma_1, \varsigma_2, \dots, \varsigma_n\}$ such that each ς_i is either a ct-sequence or of the form $\sigma \circ \varsigma$ where σ is a ct-sequence, and for any two ς_i and $\varsigma_j \in \Xi$, $\bar{f}_{\varsigma_i} \cap \bar{f}_{\varsigma_j} = \emptyset$, i.e., first action set of all sequences are all distinct from each other, then another stochastic ct-sequence $\odot_t \Xi$ is defined as:

$$\odot_t \Xi = \begin{cases} \varsigma_i, & \text{if } s \in \bar{I}_{\varsigma_i} \setminus \bigcup_{j \neq i} \bar{I}_{\varsigma_j} \\ \mu_{\Xi, t}, & \text{otherwise} \end{cases}$$

where s is the current state, and $\mu_{\Xi, t} : \Omega \times \Xi \rightarrow [0, 1]$ selects and executes one of $\varsigma_1, \dots, \varsigma_n$ according to a probability distribution based on the observed history of the last t steps. Here, $\odot_t \Xi$ behaves like ς_i if any other $\varsigma_j \in \Xi$ is not applicable at state s . $\bar{I}_{\odot_t \Xi} = \bar{I}_{\varsigma_1} \cup \dots \cup \bar{I}_{\varsigma_n}$ and $\bar{f}_{\odot_t \Xi} = \bar{f}_{\varsigma_1} \cup \dots \cup \bar{f}_{\varsigma_n}$.

4. Nothing generated by rules other than 1-3 is a ct-sequence.

Let u be a ct-sequence and v be a stochastic ct-sequence. Their combination, denoted $u \otimes_t v$, that behaves both like u and v can be defined as follows depending on the form of v :

- If v is a ct-sequence, then (i) If $u \preceq v$ (or $v \preceq u$), then $u \otimes_t v = u \cup v$ (or $v \cup u$). (ii) If $f_u \neq f_v$, i.e., their first actions are different from each other, then $u \otimes_t v = \odot_t \{u, v\}$. (iii) If $A_{u^{1, k-1}} = A_{v^{1, k-1}}$ and $a_{u, k} \neq a_{v, k}$, i.e., their action sequences have a maximal common prefix of length $k - 1$, then $u \otimes_t v = (u^{1, k-1} \cup v^{1, k-1}) \circ (\odot_{t+k} \{u_k, v_k\})$.
- if $v = \sigma \circ \varsigma$ where σ is a ct-sequence, then, (i) if $|u| < |\sigma|$ and $u \preceq \sigma$, i.e., action sequence of u is a prefix of action sequence of σ , then $u \otimes_t v = (\sigma \cup u) \circ \varsigma$ (ii) if $\sigma \preceq u$, then $u \otimes_t v = (\sigma \cup u^{1, |\sigma|}) \circ (u^{|\sigma|+1} \otimes_{t+|\sigma|+1} \varsigma)$ (iii) if $f_u \neq f_\sigma$, i.e., first actions of u and σ are different from each other, then $u \otimes_t v = \odot_t \{u, v\}$ (iv) if $A_{u^{1, k-1}} = A_{\sigma^{1, k-1}}$ and $a_{u, k} \neq a_{\sigma, k}$, i.e., action sequences of u and σ differ at a position $k \leq |\sigma|$, then $u \otimes_t v = (\sigma^{1, k-1} \cup u^{1, k-1}) \circ (\odot_{t+k} \{u^k, \sigma^k \circ \varsigma\})$
- if $v = \odot_t \{\varsigma_1, \dots, \varsigma_n\}$, then $u \otimes_t v = \odot_t \{\varsigma_1, \dots, \varsigma_{i-1}, u \otimes_t \varsigma_i, \varsigma_{i+1}, \dots, \varsigma_n\}$ if $f_u \in \bar{f}_{\varsigma_i}$, and $u \otimes_t v = \odot_t \{\varsigma_1, \dots, \varsigma_n, u\}$ otherwise.

For example, given $\sigma_{ee} = \langle [(0, 0), (4, 2)], e \rangle \langle [(0, 1), (4, 3)], e \rangle$ corresponding to the action pattern of moving east twice in 5×5 grid world, we have $\sigma_{en} \otimes_t \sigma_{ee} = \langle I_{\sigma_{en, 1}} \cup I_{\sigma_{ee, 1}}, e \rangle \circ I'$, where s is the state observed by the agent after executing first action and I' is $\langle I_{\sigma_{en, 2}}, n \rangle$ if $s \in [(0, 4), (3, 4)]$, $\langle I_{\sigma_{ee, 2}}, e \rangle$ if $s \in [(4, 1), (4, 3)]$, and either one otherwise. Depending on s , $\sigma_{en} \oplus_t \sigma_{ee}$ continues its behavior either as σ_{en} or σ_{ee} (see Figure 1).

Now, let $\prod \Sigma = \sigma_1 \otimes_0 (\sigma_2 \otimes_0 (\dots \otimes_0 (\sigma_{n-1} \otimes_0 \sigma_n) \dots))$, $\Gamma_{\eta, t}$ be the sequence of actions taken during the last t steps of history $\eta \in \Omega$, and define $\mu_{\Xi, t} : \Omega \times \Xi \rightarrow [0, 1]$ as $\mu_{\Xi, t}(\eta, \varsigma) = \max\{P(s, \sigma_i) : s \in \bar{I}_{\odot_t \Xi}\}$

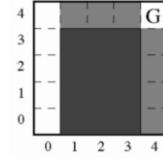


Figure 1. After the first step, $\sigma_{en} \oplus \sigma_{ee}$ behaves like σ_{en} if current state is in the right light shaded area, behaves like σ_{ee} if it is in the top light shaded area, and either as σ_{en} or σ_{ee} if it is in the dark shaded area.

$A_{\sigma_i^{1, t-1}} = \Gamma_{\eta, t-1}$ and $a_{\sigma_i, t} \in \bar{f}_\varsigma$. If $\sigma_\tau \in \Sigma$ is selected by P at state s and executed successfully $|\sigma_\tau|$ steps until termination, then initiated at s , $\prod \Sigma$ takes exactly the same actions as σ_τ , and exhibit the same behavior and therefore can be used to represent ct-sequences in Σ up to P .

We can computationally represent a stochastic ct-sequence ς , and consequently $\prod \Sigma$ defined above, in a natural form by using a rooted tree T_ς , called *sequence tree*. Each edge of the sequence tree is labeled with an action, and there is a node for every action sequence p that can be generated by ς when executed, denoted by ν_p . The root node designates the empty action sequence. Let p and q be two action sequences such that $q = pa$ where $a \in A$, then ν_p is connected to ν_q by an edge with label a . Furthermore, ν_q is associated with an initiation set I_{ν_q} specifying the states where action a can be selected after the execution of action sequence p . A node with $k > 1$ outgoing edges is called a decision point of order k . Let E_ν be the set of labels on out-going edges of ν . If ν is a decision point then ν is assigned a branching condition $\vartheta_\nu : \Omega \times E_\nu \rightarrow [0, 1]$ which selects one of the actions in E_ν based on the observed history. Given a stochastic ct-sequence ς , its corresponding sequence tree can be constructed in linear time recursively by creating and connecting decision points starting from the root node, based on the structure of ς .

Initiated at state s , a sequence tree T_ς selects actions by following a path starting from the root node and choosing edges according to initiation sets and branching conditions of nodes. Let $\widehat{\nu}$ denote the active node of T_ς , which is initially the root node, $e_{u, v}$ denotes the action on the label of the edge connecting u to v , and s be the current state. Let $N = \{\nu_1, \dots, \nu_k\}$ be the set of child nodes of $\widehat{\nu}$ with initiation sets containing s . At each step, if $N = \emptyset$, then execution of T_ς terminates. If $N = \{\nu_i\}$, then action $e_{\widehat{\nu}, \nu_i}$ is selected and $\widehat{\nu}$ is set to ν_i , or else, action a is selected using probability distribution $\vartheta_{\widehat{\nu}}(\eta, \cdot)$ and $\widehat{\nu}$ is set to ν_i such that $e_{\widehat{\nu}, \nu_i} = a$.

When the set of ct-sequences, Σ , is known in advance, a stochastic ct-sequence $\prod \Sigma$ and corresponding sequence tree $T_{\prod \Sigma}$ can be constructed and utilized instead of sequences in Σ . However, if such an information is not available prior to learning, it is still possible to discover useful ct-sequences online, as learning progresses, and iteratively build a sequence tree to represent them based on histories observed by the agent as we describe in the next section.

4 Discovering Ct-sequences

A history that starts with state s and obtained by following a policy π until the end of an episode (or between designated conditions such as when a reward peak is reached) is called a π -history of s . Let π' and π^* denote the agent's current policy and an optimal policy, respectively, and $h' = s_1 a_1 r_2 \dots r_t s_t$ be a π' -history of length t for state s_1 . Total cumulative reward of h' is defined as $r(h') = r_2 + \gamma r_3 + \dots + \gamma^{t-1} r_t$. Now, suppose that in h' a state appears at two positions i and j , i.e., $s_i = s_j, i \neq j$; and consider

the sequence $h'' = s_1 a_1 r_2 \dots r_i s_i a_{j+1} r_{j+1} \dots r_t s_t$, where s_i and s_j are united and the sequence in between is removed. h'' is also a (synthetic) π' -history for state s_1 and could be a better candidate for being a π^* -history if $r(h'') > r(h')$. Every suffix of h' of the form $h'_i = s_i a_i r_{i+1} \dots r_t s_t$ for $i = 2..t - 1$ is also a π' -history. Combining these two observations, we can generate a set of π^* -history candidates by processing h' from back to front. Let bh'_s denote the π' -history for state s with maximum total cumulative reward; initially $bh'(s_{t-1}) = s_{t-1} a_{t-1} r_{t-1} s_t$. For each $s_i, i = t - 2..1$, if s_i is not encountered before (i.e., for $j > i, s_j \neq s_i$) or $r_i + \gamma r(bh'_{s_{i+1}})$ is higher than the total cumulative reward of the current bh'_{s_i} , $r(bh'_{s_i})$, then bh'_{s_i} is replaced by $s_i a_i r_{i+1} b h'_{s_{i+1}}$. Finally, for each unique s_i in (s_1, \dots, s_t) , resulting bh'_{s_i} is used as a probable π^* -history for state s_i . For every π' -history $h' = s_1 a_1 r_2 \dots r_t s_t$, there is a corresponding ct-sequence $\sigma_{h'} = \langle \{s_1\}, a_1 \rangle \dots \langle \{s_{t-1}\}, a_{t-1} \rangle$. Let H_i be the set of such ct-sequences generated at i^{th} episode. As learning progresses, ct-sequences with action sequences that are part of useful subpolicies, and their successors, would appear more frequently in the recent H_i 's, whereas occurrence rate of ct-sequences with dominated subpolicies or that are less general would be low. Therefore, by keeping track of such ct-sequences and generalizing them, one can identify valuable abstractions and utilize them to improve learning performance. This can be accomplished efficiently using a modified version of a sequence tree.

An *extended* sequence tree is a sequence tree which has additional eligibility and expected reward attributes. Let p and q be two action sequences such that $q = pa$ where $a \in A$, then ν_p is connected to ν_q by an edge $e_{p,q}$ with label $\langle a, \psi \rangle$ where ψ denotes the eligibility of $e_{p,q}$, an indicator of how frequent action sequence q is executed, and ν_q holds a list of tuples $\langle s_1, \xi_{s_1}, R_{s_1} \rangle, \dots, \langle s_k, \xi_{s_k}, R_{s_k} \rangle$ stating that a can be selected at s_i after executing p with an expected total cumulative reward of R_{s_i} . ξ_{s_i} is the eligibility of s_i at ν_q , which indicates how frequent a is actually selected at state s_i after p is executed. $I_{\nu_p} = \{s_1, \dots, s_k\}$ is the initiation set of ν_p . A π -history $h = s_1 a_1 r_2 \dots r_t s_t$ can be added to an extended sequence tree T by starting from the root node and following edges according to their labels. Let \hat{n} denote the active node of T , which is initially the root node. For $i = 1..t - 1$, if there is a node n such that \hat{n} is connected to n by an edge $e_{\hat{n},n} = \langle a_i, \psi_{\hat{n},n} \rangle$, then (i) $\psi_{\hat{n},n}$ is incremented. (ii) if n contains a tuple $\langle s_i, \xi_{s_i}, R_{s_i} \rangle$, then ξ_{s_i} is incremented and R_{s_i} is updated if $r(h_i) > R_{s_i}$, otherwise a new tuple $\langle s_i, 1, r(h_i) \rangle$ is added to n , and (iii) \hat{n} is set to n afterwards. If such a node n does not exist, a new node with tuple $\langle s_i, 1, r(h_i) \rangle$ is created, and \hat{n} is connected to this node by an edge $e = \langle a_i, 1 \rangle$; the new node becomes the active node.

After each episode, based on the sequence of states, actions and rewards observed during the episode (or between two termination conditions in case of non-episodic tasks) a set of probable π^* -histories are generated and added to the path tree. The eligibility values of edges are decremented by a factor of $0 < \psi_{decay} < 1$, and eligibility values in the tuples of each node are decremented by a factor of $0 < \xi_{decay} \leq 1$. For an action sequence σ that is frequently used, edges on the path from the root node to the node representing σ and tuples with visited states would have high eligibility values, whereas they would decay to 0 for sequences that are used less. Hence, in order to filter traces of previous policies that are no longer useful, tuples with eligibility values less than a given threshold, $\xi_{threshold}$, are removed from the nodes, and each node with empty tuple list is removed from the tree together with incoming edge and the subtree rooted at it. Similarly, edges with eligibility value less than a given threshold $\psi_{threshold}$, and all connected nodes are also pruned. The

resulting extended sequence tree after performing these operations represents recent potential ct-sequences in a compact form. Since branching conditions are not available in an extended sequence tree, action selection procedure previously defined for sequence trees must be slightly modified; At node n if multiple children nodes ν_1, \dots, ν_k contain a tuple $\langle s, \xi_{s,\nu_i}, R_{s,\nu_i} \rangle$ for the current state s , then one of them is selected on the basis of R_{s,ν_i} and/or ξ_{s,ν_i} . Since ct-sequences represented by the resulting tree are constantly updated, it is not feasible to directly integrate them into RL framework by extending the value functions and Bellman equations to include options. Instead, a flat policy approach can be used by triggering the execution of the tree (and consequently represented ct-sequences) with a given probability and reflecting the action selections to underlying RL algorithm for value function updates.

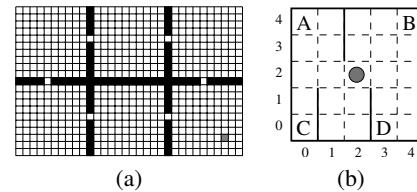


Figure 2. (a) Six-room maze, and (b) Dietterich's taxi domain.

5 Experiments

We applied the method described in Section 4 to regular Q-learning on two test domains, a six-room maze and Dietterich's taxi problem [2], and compared its performance with other RL algorithms. In all cases, the initial Q-values are set to 0, the learning rate and discount factor are $\alpha = 0.125$, and $\gamma = 0.9$, respectively. ϵ -greedy action selection mechanism is used with $\epsilon = 0.1$. In the six-room maze problem, the agent's task is to move from a randomly chosen position in the top left room to the grey shaded goal location in the bottom right room. The primitive actions are movement to a neighboring cell in four directions (Figure 2a). Each action succeeds with probability 0.9 or moves the agent perpendicular to the desired direction with probability 0.1. The agent only receives a reward of 1 when it reaches the goal location. For all other cases, it receives a small negative reward of -0.01 . Therefore, it must learn to reach the goal state as quickly as possible to maximize total discounted reward. In this task, for a better performance the agent needs to learn to make use of the passages connecting the rooms, which are the bottleneck states. Figure 3 shows sliding window (20 episodes) average of number of steps for reaching the goal averaged over 30 runs. ψ_{decay} , ξ_{decay} , and eligibility thresholds are 0.95, 0.99, and 0.01, respectively. Sequence tree is processed with a probability of 0.3 to run represented option. While processing the tree, Q-values are updated using the update rule of Q-learning on the basis of selected actions and observed states. At decision points, actions are chosen ϵ -greedily based on rewards of the tuples. For $Sarsa(\lambda)$ algorithm, λ is taken as 0.98. Used in the *macro-q* algorithm are predefined options which move the agent from any cell in a room, except bottom right one which contains the goal location, to one of two doorways that connect to neighboring rooms in minimum number of steps. As can be seen from Figure 3, Q-learning with sequence tree outperforms other algorithms and learns faster. This shows that the method is successful in discovering and utilizing ct-sequences that are indeed useful for the solution.

In the taxi problem, a taxi agent moves around on a 5×5 grid world, containing obstacles that limit the movement (Figure 2b), and tries to transport passengers located at predefined locations, to either

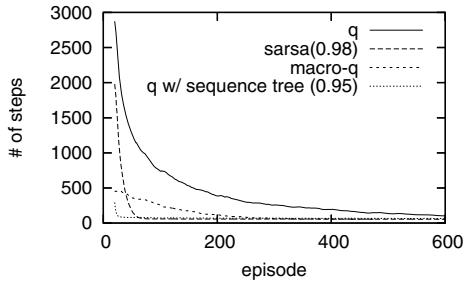


Figure 3. Results for six-room maze problem.

the same location or to another one. At each time step, the agent can either move one square in one of four directions, attempt to *pickup* a passenger, or attempt to *drop-off* the passenger being carried. If a move action would cause the agent to hit a wall, the position of the agent does not change. The movement actions are non-deterministic and with 0.2 probability agent may move perpendicular to the desired direction. Passengers can not be co-located at the same position but their destinations can be the same. There is an immediate reward of +20 if a passenger is transported successfully, -10 if pickup or drop-off actions are executed incorrectly, and -1 for any other action. An episode ends when all passengers are transported to their destinations. In order to maximize the overall cumulative reward, the agent must transport the passenger using minimum number of actions. Initial position of the taxi agent, locations and destinations of the passengers are selected randomly with uniform probability. Compared to six-room maze, taxi domain has a larger state space and possesses a hierarchical structure with repeated subtasks. Results for the taxi problem with one and two passengers are plotted in Figure 4. For the *macro-q* algorithm, predefined options that move the agent from any position to one of predefined locations in minimum number of steps are used. Various ψ_{decay} values are tested, other parameters are the same as in maze problem. Q-learning with sequence tree performs better than all other RL algorithms. The reason that it performs better than *macro-q* is the fact that *macro-q* needs to learn which options are optimal whereas Q-learning with sequence tree starts to use sub-optimal sequences immediately. Note that the convergence rate of algorithms that do not make use of abstractions decrease as common subtasks become more prominent. The results also show that the proposed method is effective in identifying solutions in such cases. Figure 5 shows the size of the sequence tree in terms of the number of nodes for different values of ψ_{decay} . Although their performances are similar, sequence tree shrinks considerably as ψ_{decay} decreases.

6 Conclusions

We emphasized the usefulness of discovering Semi-Markov options automatically using domain information acquired during learning. Further, we demonstrated the importance of constructing a dynamic and compact sequence-tree from histories. Experiments highlighted the effectiveness of utilizing such a tree in the learning process. Our future work will examine guaranteed convergence of the proposed method and its adaptation to larger domains, possibly involving continuous state variables.

ACKNOWLEDGEMENTS

This work was supported by the Scientific and Technological Research Council of Turkey under Grant No. 105E181(HD-7).

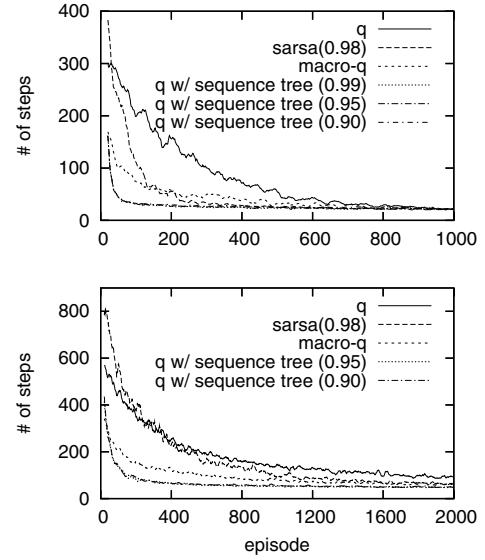


Figure 4. Results for taxi problem. One passenger (top) and two passengers (bottom).

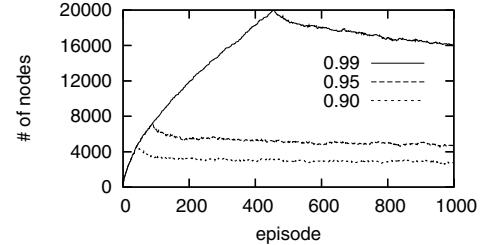


Figure 5. Size of the sequence tree for different values of ψ_{decay} .

REFERENCES

- [1] Andrew G. Barto and Sridhar Mahadevan, ‘Recent advances in hierarchical reinforcement learning’, *Discrete Event Dynamic Systems*, **13**(4), 341–379, (2003).
- [2] Thomas G. Dietterich, ‘Hierarchical reinforcement learning with the MAXQ value function decomposition’, *Journal of AI Research*, **13**, 227–303, (2000).
- [3] Michael Littman Leslie Kaelbling and Andrew Moore, ‘Reinforcement learning: A survey’, *Journal of AI Research*, **4**, 237–285, (1996).
- [4] Amy McGovern, *Autonomous Discovery of Temporal Abstractions From Interactions With An Environment*, Ph.D. dissertation, University of Massachusetts Amherst, May 2002.
- [5] Amy McGovern and Andrew G. Barto, ‘Automatic discovery of subgoals in reinforcement learning using diverse density’, in *Proc. of Int. Conf. on Machine Learning*, pp. 361–368, (2001).
- [6] Ishai Menache, Shie Mannor, and Nahum Shimkin, ‘Q-cut - dynamic discovery of sub-goals in reinforcement learning’, in *Proc. of European Conf. on Machine Learning*, pp. 295–306, (2002).
- [7] Özgür Şimşek, Alicia P. Wolfe, and Andrew G. Barto, ‘Identifying useful subgoals in reinforcement learning by local graph partitioning’, in *Proc. of Int. Conf. on Machine Learning*, (2005).
- [8] Martin Stolle and Doina Precup, ‘Learning options in reinforcement learning’, in *Proc. of Int. Symposium on Abstraction, Reformulation and Approximation*, pp. 212–223, (2002).
- [9] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.
- [10] Richard S. Sutton, Doina Precup, and Satinder Singh, ‘Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning’, *Artificial Intelligence*, **112**(1-2), 181–211, (1999).

Least Squares SVM for Least Squares TD Learning

Tobias Jung¹ and Daniel Polani²

Abstract. We formulate the problem of least squares temporal difference learning (LSTD) in the framework of least squares SVM (LS-SVM). To cope with the large amount (and possible sequential nature) of training data arising in reinforcement learning we employ a subspace based variant of LS-SVM that sequentially processes the data and is hence especially suited for online learning. This approach is adapted from the context of Gaussian process regression and turns the unwieldy original optimization problem (with computational complexity being cubic in the number of processed data) into a reduced problem (with computational complexity being linear in the number of processed data). We introduce a QR decomposition based approach to solve the resulting generalized normal equations incrementally that is numerically more stable than existing recursive least squares based update algorithms. We also allow a forgetting factor in the updates to track non-stationary target functions (i.e. for the use with optimistic policy iteration). Experimental comparison with standard CMAC function approximation indicate that LS-SVMs are well-suited for online RL.

1 Introduction

Least-squares TD (LSTD) One very important subproblem in reinforcement learning (RL) is the policy evaluation problem, e.g. see [13]. The goal is to compute the value function under a fixed policy π (we assume a deterministic π), which is taken to be the infinite horizon, discounted sum of rewards $V^\pi(\mathbf{x}) = E^\pi \{ \sum_{i=0}^{\infty} \gamma^i r_i \mid \mathbf{x}_0 = \mathbf{x} \}$ for the Markov chain resulting from starting in state \mathbf{x} and choosing actions according to policy π afterwards. Here r_i denotes the reward obtained while traversing from state \mathbf{x}_{i-1} to \mathbf{x}_i , and $\gamma \in [0, 1]$ denotes a discount factor. The expectation E^π is meant wrt. the distribution of complete trajectories. However, the Bellman equation allows us to write V^π just using the distribution of the successor states \mathbf{x}' , i.e. $V^\pi(\mathbf{x}) = E^\pi \{ r_t + \gamma V^\pi(\mathbf{x}') \mid \mathbf{x}_{t-1} = \mathbf{x} \}$. To compute V^π from actual trajectories we can employ temporal difference learning, e.g. TD(0), which employs *incremental* updates of the form $[r + \gamma V(\mathbf{x}')] - V(\mathbf{x})$ for the observed transition from \mathbf{x} to \mathbf{x}' as an unbiased estimate for the true (unknown) expectation. It works well in conjunction with parametrized function approximation and is usually trained with (stochastic) gradient descent.

A variant that computes a solution in closed form is LSTD [2, 8], minimizing the Bellman residuals for *all* observed transitions in a least squares sense. This approach is more efficient since it makes simultaneous use of all transitions seen so far. One disadvantage³

¹ University of Mainz, Germany, email: tjung@informatik.uni-mainz.de

² University of Hertfordshire, UK, email: d.polani@herts.ac.uk

³ Also, if the transitions are stochastic, we need a second independent sample for each transition [1]. To allow a clearer exposition of our algorithm we will only consider deterministic transitions. For stochastic MDPs this problem can be side-stepped by least squares fixed point approximation [2, 8].

of LSTD is that it involves solving (recursively) the corresponding normal equations which is more costly than mere gradient descent and has limited its use to parametrized function approximation.

In this paper we aim to exploit the data efficiency of LSTD and the superior generalization capabilities of kernel-based methods as underlying function approximator. Kernel-based methods (e.g. SVM) are a flexible and more powerful alternative to parametric methods, since the solution does not depend on a small number of features chosen beforehand but is expanded in the training data itself. In the past, kernel-based methods have mostly been used for batch RL [4]. More recently, based on the online sparsification for Gaussian process regression (GPR) from [3], a GPR approach for online TD learning was proposed in [5].

Our approach is in some respects similar to theirs (since generally the predictor obtained from LS-SVM equals the posterior mean of the GPR and the regularization parameter just corresponds to the noise estimate). Our contribution lies 1.) in an alternative, elegant derivation using LS-SVM, 2.) presenting a QR decomposition based update scheme that is numerically more stable, 3.) including a forgetting factor to track non-stationary target functions (i.e. for the use with optimistic policy iteration) and 4.) providing further experimental evidence that kernel-based learning is well-suited for RL.

LS-SVM for LSTD Assume we have observed the t deterministic transitions $\{(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i, r_i)\}_{i=1}^t$, $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$, $r_i \in \mathcal{Y} \subset \mathbb{R}$. The goal is to find a function $V : \mathcal{X} \rightarrow \mathcal{Y}$ which minimizes the residuals for the given data and generalizes well to unseen data. We proceed in the usual way: choose the reproducing kernel Hilbert space (RKHS) \mathcal{H} of functions $V : \mathcal{X} \rightarrow \mathcal{Y}$ endowed with kernel k , where $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y}$ is a symmetric, positive function (e.g. think of Gaussian radial basis functions). To learn the unknown function we solve a Tikhonov functional of the special form

$$\min_{V \in \mathcal{H}} \frac{1}{t} \sum_{i=1}^t \left[(V(\mathbf{x}_{i-1}) - \gamma V(\mathbf{x}_i)) - r_i \right]^2 + \frac{\Lambda}{t} \|V\|_{\mathcal{H}}^2. \quad (1)$$

The first term measures the error in the approximation (of temporal differences) and the second term penalizes the complexity (i.e. the roughness) of the candidate V . The regularization parameter Λ/t controls the trade-off between them. The Representer theorem tells us that every solution to (1) is the sum of kernels centered on the data: i.e. $V(\cdot) = \sum_{i=1}^t \alpha_i k(\mathbf{x}_i, \cdot)$ where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_t)^T$ are the coefficients that we need to determine. Define a $t \times (t+1)$ band matrix \mathbf{D} by setting $d_{ii} = 1$ and $d_{i,i+1} = -\gamma$. Replacing V in (1) by its expansion and using that in RKHS we have the property $\langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \rangle_{\mathcal{H}} = k(\mathbf{x}_i, \mathbf{x}_j)$, we then arrive at

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^t} \frac{1}{t} \|\mathbf{DK}_{tt} \boldsymbol{\alpha} - \mathbf{r}\|^2 + \frac{\Lambda}{t} \boldsymbol{\alpha}^T \mathbf{K}_{tt} \boldsymbol{\alpha} \quad (2)$$

with \mathbf{K}_{tt} being the $t \times t$ kernel matrix $[\mathbf{K}_{tt}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{r} = (r_1, \dots, r_t)^T$ being the sequence of observed rewards. Solving for $\boldsymbol{\alpha}$ we obtain

$$(\mathbf{H}_{tt}^T \mathbf{H}_{tt} + \Lambda \mathbf{K}_{tt}) \boldsymbol{\alpha} = \mathbf{H}_{tt}^T \mathbf{r}. \quad (3)$$

where the $t \times t$ matrix \mathbf{H}_{tt} is defined as $\mathbf{H}_{tt} =_{\text{def}} \mathbf{D} \mathbf{K}_{tt}$.

The central difficulty when using a non-parametric approach like (1) is that the computational and memory complexity scales badly in the data: the kernel matrix is dense and requires $\mathcal{O}(t^2)$ storage, solving the generalized normal equations (3) requires $\mathcal{O}(t^3)$ operations and even every single prediction using the learned model requires $\mathcal{O}(t)$ operations. Clearly, this would be completely infeasible when used in conjunction with RL. In the next section we review *sparse approximation* as means to overcome this problem.

2 Background: Projection on a subset of kernels

The technique of sparse approximation attempts to approximate the kernel matrix using only a small subset of the data and avoids explicit use and storage of the full $t \times t$ kernel matrix. In the subsequent computations it allows us to consider a greatly reduced number of variables. It is a very common technique in the context of Gaussian process regression [3, 5, 11, 14].

Sparse approximation Sparse approximation is based on the observation that the kernel matrix often has rapidly decaying eigenvalues. Vanishing eigenvalues indicate that the data spans – approximately – a rather low dimensional manifold in the feature space. Thus, instead of using all the data points we can restrict ourselves to use only those composing a basis of this manifold and project the remaining ones onto their span. Let this basis be $\{k(\tilde{\mathbf{x}}_i, \cdot)\}_{i=1}^m$ for a subset of m chosen data points⁴ ($m \leq t$). Now, for arbitrary \mathbf{x}_i kernel $k(\mathbf{x}_i, \cdot)$ is approximated using only the m chosen basis elements

$$k(\mathbf{x}_i, \cdot) \approx \sum_{j=1}^m a_{ij} k(\tilde{\mathbf{x}}_j, \cdot) \quad i = 1 \dots t. \quad (4)$$

In particular \mathbf{a}_i becomes the j -th unit vector if \mathbf{x}_i equals basis element $\tilde{\mathbf{x}}_j$ for some index j , so that the m data points constituting the basis are represented exactly in the feature space. The remaining $t - m$ ones are represented only approximately. To obtain the coefficients a_{ij} we minimize in RKHS \mathcal{H} the distance $d_i := \|k(\mathbf{x}_i, \cdot) - \sum_{j=1}^m a_{ij} k(\tilde{\mathbf{x}}_j, \cdot)\|_{\mathcal{H}}^2$. Writing this in terms of the inner product and setting its derivative to zero we arrive at

$$\mathbf{a}_i = \mathbf{K}_{mm}^{-1} \mathbf{k}_i \quad (5)$$

where \mathbf{K}_{mm} is the $m \times m$ kernel matrix corresponding to the basis elements with $[\mathbf{K}_{mm}]_{ij} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$ and $\mathbf{k}_i = (k(\tilde{\mathbf{x}}_1, \mathbf{x}_i), \dots, k(\tilde{\mathbf{x}}_m, \mathbf{x}_i))^T$. Once we have determined \mathbf{a}_i for input \mathbf{x}_i , we obtain the corresponding approximation error as

$$d_i = k_{ii} - \mathbf{k}_i^T \mathbf{a}_i \quad (6)$$

where $k_{ii} = k(\mathbf{x}_i, \mathbf{x}_i)$. This quantity tells us how well the basis is able to approximate the i -th data point.

⁴ To avoid using cumbersome index sets we just mark data points selected into the basis by a tilde

Online selection of the subset Next we need to deal with the selection of the subset $\{\tilde{\mathbf{x}}_i\}_{i=1}^m$ which is going to make up our basis. Possibilities include a greedy and iterative approach, adding the element to the basis that best represents all remaining non-basis elements [12], the incomplete Cholesky factorization [7] or just random selection [14]. However, all methods but the latter are unsuitable when the data arrives incrementally. For this particular purpose, [3] and later [6] have proposed a greedy online scheme that builds up the basis incrementally and expands it whenever it becomes necessary. We will apply their method in our approach without modification: assume the data arrives sequentially at $t = 1, 2, \dots$ and that we can examine every data point once but cannot revise our decision later. We maintain a list of currently chosen basis-elements \mathcal{D}_m (the dictionary), the inverse \mathbf{K}_{mm}^{-1} and assume that after seeing $t - 1$ samples dictionary \mathcal{D}_m contains m elements. When a new example \mathbf{x}_t arrives we compute how well it can be approximated using \mathcal{D}_m , i.e. we compute (6). If this distance is below some chosen threshold ν , the current basis represents \mathbf{x}_t well enough and we do not need to consider it further. If the distance exceeds the chosen tolerance, we add $k(\mathbf{x}_t, \cdot)$ to the basis: expand $\mathcal{D}_{m+1} = \mathcal{D}_m \cup \{\mathbf{x}_t\}$ and update \mathbf{K}_{mm}^{-1} . Since the change to \mathbf{K}_{mm} just involves appending \mathbf{k}_t , the new inverse can be obtained recursively either using the partitioned matrix inversion formula or the Cholesky factorization. Computational cost and storage is $\mathcal{O}(m^2)$.

It can be shown [6] that for threshold $\nu > 0$ the number m of selected elements is bound by $\text{Const} \cdot \nu^{-d}$, where d is the dimensionality of the input domain $\mathcal{X} \subset \mathbb{R}^d$. Hence, for a continuous stream of data $\{\mathbf{x}_t\}_{t=1}^\infty$ the dictionary remains finite. For any reasonable choice of ν (e.g. $\nu = 10^{-2}$) the number of selected elements m is usually only a fraction of the number t of observed data points and further stops increasing during the later stages of learning.

Solving a reduced problem Distinguish between batch and online selection of the subset. First consider the batch case, where we compute the \mathbf{a}_i after we have determined the m basis elements. Define a $t \times m$ matrix \mathbf{A} with rows \mathbf{a}_i^T from (5). Then we can write

$$\mathbf{A}^T = \mathbf{K}_{mm}^{-1} \mathbf{K}_{tm}^T \quad (7)$$

with \mathbf{K}_{tm} being the m columns of the full kernel matrix corresponding to the m indices of the basis elements. We obtain that $\hat{\mathbf{K}}_{tt} =_{\text{def}} \mathbf{A} \mathbf{K}_{mm} \mathbf{A}^T = \mathbf{K}_{tm} \mathbf{K}_{mm}^{-1} \mathbf{K}_{tm}^T$ is a low-rank approximation of the full kernel matrix \mathbf{K}_{tt} . To benefit from this approximation [12] suggest the following: solve the regularized risk functional (1) using all data points, but allow only the coefficients of the m points constituting the basis to have non-zero values. This leads to a modified problem (c.f. the original problem (2)):

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \frac{1}{t} \|\mathbf{D} \mathbf{K}_{tm} \boldsymbol{\alpha} - \mathbf{r}\|^2 + \frac{\Lambda}{t} \boldsymbol{\alpha}^T \mathbf{K}_{mm} \boldsymbol{\alpha} \quad (8)$$

and to the system of linear equations

$$(\mathbf{H}_{tm}^T \mathbf{H}_{tm} + \Lambda \mathbf{K}_{mm}) \boldsymbol{\alpha} = \mathbf{H}_{tm}^T \mathbf{r} \quad (9)$$

where the $t \times m$ matrix \mathbf{H}_{tm} is defined as $\mathbf{H}_{tm} = \mathbf{D} \mathbf{K}_{tm}$. Thus we end up solving the $m \times m$ system (9) rather than the $t \times t$ system (3), which reduces computational and storage costs to manageable $\mathcal{O}(m^3)$ and $\mathcal{O}(m^2)$ respectively.

If we assume online selection of the basis elements, to compute the \mathbf{a}_i we can use only the basis elements found up to the current time step (whereas in the batch case we would use the complete basis). Indices in \mathbf{a}_i corresponding to elements added in future are set to

zero. Instead of (7) we obtain that $\tilde{\mathbf{K}}_{tm} =_{\text{def}} \mathbf{A}\mathbf{K}_{mm}$ is only an approximation of the true submatrix \mathbf{K}_{tm} . To indicate this difference, we will use the notation $\tilde{\mathbf{H}}_{tm}$ instead of \mathbf{H}_{tm} in (9).

3 Time recursive solution using the QR method

The usual way to solve the normal equations (9) incrementally for each observed transition $(\mathbf{x}_{t-1} \rightarrow \mathbf{x}_t, r_t)$ is to employ recursive least squares methods. Our approach is different though, we present a QR based solution adapted from the adaptive filtering literature, e.g. [10], which comes at the same cost of $\mathcal{O}(m^2)$ operations per step but has increased numerical stability.

Concerning notation: the first subscript t will denote the current number of transitions and the second subscript m will denote the current number of basis elements. Substituting in (9) the $m \times m$ cross-product matrix by $\Phi_{tm} = (\tilde{\mathbf{H}}_{tm}^T \tilde{\mathbf{H}}_{tm} + \Lambda \mathbf{K}_{mm})$, and the rhs by $m \times 1$ vector $\mathbf{s}_{tm} = \tilde{\mathbf{H}}_{tm}^T \mathbf{r}$ eq. (9) becomes

$$\Phi_{tm} \alpha_t = \mathbf{s}_{tm} \quad (10)$$

Next we introduce the Cholesky factorization $\Phi_{tm} = \Phi_{tm}^{1/2} \Phi_{tm}^{T/2}$ and an auxiliary $m \times 1$ vector \mathbf{q}_{tm} . To solve (10) we first solve $\Phi_{tm}^{1/2} \mathbf{q}_{tm} = \mathbf{s}_{tm}$ and then $\Phi_{tm}^{T/2} \alpha_t = \mathbf{q}_{tm}$.

Assume we have observed $t - 1$ transitions and selected m basis functions. Each update t now proceeds as follows: we check if \mathbf{x}_t is represented by the current basis well enough or if we will need to augment the basis. The involved quantities $\{\Phi_{tm}^{1/2}, \mathbf{q}_{tm}, \mathbf{K}_{mm}^{1/2}\}$ are then updated accordingly. At the end we just solve $\Phi_{tm}^{T/2} \alpha_t = \mathbf{q}_{tm}$ to obtain the desired α_t . Below we sketch the derivation of the central steps (see Fig. 1 for the complete algorithm).

Step 1: Processing new data without augmenting the basis: \mathbf{x}_t is approximately represented by the current dictionary, therefore we only add the row vector $\mathbf{h}_t^T = (\mathbf{k}_{t-1} - \gamma \mathbf{k}_t)^T$ to the designmatrix. Thus,

$$\begin{aligned} \Phi_{tm} &= \left[\begin{array}{c} \tilde{\mathbf{H}}_{t-1,m} \\ \mathbf{h}_t^T \end{array} \right]^T \left[\begin{array}{c} \tilde{\mathbf{H}}_{t-1,m} \\ \mathbf{h}_t^T \end{array} \right] + \Lambda \mathbf{K}_{mm} \\ &= \Phi_{t-1,m} + \mathbf{h}_t \mathbf{h}_t^T \end{aligned}$$

and hence

$$\Phi_{tm} \mathbf{q}_{tm} = \Phi_{t-1,m} \mathbf{q}_{t-1,m} + r_t \mathbf{h}_t.$$

To carry out this update we employ the *array* notation from [10]: we start by arranging the quantities $\{\Phi_{t-1,m}^{1/2}, \mathbf{q}_{t-1,m}\}$ from the previous step and the newly obtained information $\{\mathbf{h}_t, r_t\}$ in a clever way. If we then generate a orthogonal transformation Θ_t to lower triangulize the resulting $(m+1) \times (m+1)$ array

$$\left[\begin{array}{cc} \Phi_{t-1,m}^{1/2} & \mathbf{h}_t \\ \mathbf{q}_{t-1,m} & r_t \end{array} \right] \Theta_t = \left[\begin{array}{cc} \Phi_{tm}^{1/2} & \mathbf{0} \\ \mathbf{q}_{tm} & * \end{array} \right] \quad (11)$$

we can directly read off the desired quantities $\{\Phi_{tm}^{1/2}, \mathbf{q}_{tm}\}$ from the right hand side⁵ of (11) due to the norm and inner product preserving properties of orthogonal transformations. To determine Θ_t we can, for example, apply m Givens rotations to successively annihilate the rightmost column \mathbf{h}_t in (11).

⁵ The * denotes a scalar quantity that is not relevant for us.

Step 2: Processing new data and augmenting the basis: \mathbf{x}_t is not approximately represented by the current dictionary. Now we actually have to perform two substeps. The first substep is to account for the change when adding a new example (which adds the new row $\mathbf{h}_t^T = (\mathbf{k}_{t-1} - \gamma \mathbf{k}_t)^T$ to the designmatrix and is step 1 from above) and the second substep is to account for the change when adding a new basis function (which adds the new column $\tilde{\mathbf{H}}_{t-1,m} \mathbf{a}_t$ to the designmatrix). Setting $h_{tt} = k(\mathbf{x}_{t-1}, \mathbf{x}_t) - \gamma k(\mathbf{x}_t, \mathbf{x}_t)$ we obtain the recursion

$$\begin{aligned} \Phi_{t,m+1} &= \left[\begin{array}{cc} \tilde{\mathbf{H}}_{t-1,m} & \tilde{\mathbf{H}}_{t-1,m} \mathbf{a}_t \\ \mathbf{h}_t & h_{tt} \end{array} \right]^T \left[\begin{array}{cc} \tilde{\mathbf{H}}_{t-1,m} & \tilde{\mathbf{H}}_{t-1,m} \mathbf{a}_t \\ \mathbf{h}_t & h_{tt} \end{array} \right] \\ &\quad + \Lambda \left[\begin{array}{cc} \mathbf{K}_{mm} & \mathbf{k}_t \\ \mathbf{k}_t^T & k_{tt} \end{array} \right] \\ &= \left[\begin{array}{cc} \Phi_{tm} & \Phi_{t-1,m} \mathbf{a}_t + h_{tt} \mathbf{h}_t \\ \mathbf{a}_t^T \Phi_{t-1,m}^T + h_{tt} \mathbf{h}_t^T & \zeta \end{array} \right] \end{aligned}$$

where ζ is short for $\zeta = \mathbf{a}_t^T \Phi_{t-1,m} \mathbf{a}_t + h_{tt}^2 + \Lambda(k_{tt} - \mathbf{a}_t^T \mathbf{k}_t)$. After computing ζ (where we need $\Phi_{t-1,m}$) we perform Step 1 (above) to obtain Φ_{tm} . Then we update the Cholesky factor $\Phi_{tm}^{1/2}$ by solving $\Phi_{tm}^{1/2} \mathbf{u} = \Phi_{t-1,m} \mathbf{a}_t + h_{tt} \mathbf{h}_t$ for \mathbf{u} and setting scalar $\beta = \sqrt{\zeta - \mathbf{u}^T \mathbf{u}}$. Finally we obtain the desired quantities $\{\Phi_{t,m+1}^{1/2}, \mathbf{q}_{t,m+1}\}$ via

$$\begin{aligned} \Phi_{t,m+1}^{1/2} &= \left[\begin{array}{cc} \Phi_{tm}^{1/2} & \mathbf{0} \\ \mathbf{u}^T & \beta \end{array} \right] \\ \mathbf{q}_{t,m+1} &= \left[\begin{array}{c} \mathbf{q}_{tm} \\ (\mathbf{a}_t^T \Phi_{t-1,m}^{1/2} \mathbf{q}_{t-1,m} + h_{tt} r_t - \mathbf{u}^T \mathbf{q}_{tm}) / \beta \end{array} \right] \end{aligned}$$

Exponential forgetting Our contribution to track a non-stationary target function (e.g. when using this approach for optimistic policy iteration and slowly changing policies π) is to include a forgetting factor $0 \ll \lambda < 1$ (e.g. $\lambda = 0.999$) that puts exponentially less emphasis on past experiences. Instead of (9) we solve a modified problem with $\Sigma_t =_{\text{def}} \text{diag}(\lambda^{t-1}, \dots, \lambda, 1)$

$$(\tilde{\mathbf{H}}_{tm}^T \Sigma_t \tilde{\mathbf{H}}_{tm} + \Lambda \lambda^t \mathbf{K}_{mm}) \alpha = \tilde{\mathbf{H}}_{tm}^T \Sigma_t \mathbf{r}$$

The derivation is similar to above, and the resulting updates are summarized in the complete algorithm in Fig. 1.

4 Experiments

Here we use simulation experiments both for policy evaluation (offline learning) and optimal control (online learning) to examine if our approach indeed benefits from better generalization when compared with two standard parametric methods: (1) grid-based tilecoding (CMAC) and (2) radial basis function networks.

Policy evaluation The first experiment is a puddle-world consisting of 101x101 cells (scaled to [0,1]). Possible actions are moving into one of the four adjacent cells. The goal state is the cell in the upper right corner. Each step not leading into the goal yields the reward $r = -0.1$. Three overlapping "puddles" (Gaussians) are placed at random and incur an additional penalty proportional to their activation. The location and spread of the puddles is depicted in Fig. 3. We computed the optimal value function and policy for this domain and generated transitions for 500 different episodes, each starting from a randomly chosen state and following the optimal policy to the goal (resulting in a training set of $\sim 50,000$ observed transitions). We then

Algorithm Online policy evaluation with sparse LS-SVM and exponential forgetting

Store: Dictionary $\mathcal{D}_m = \{\tilde{\mathbf{x}}_i\}_{i=1}^m, \Phi_{tm}^{1/2}, \mathbf{q}_{tm}, \mathbf{K}_{mm}^{1/2}$, current number of basis functions m

Parameter: Accuracy ν , regularization Λ , discount factor γ , forgetting factor λ

Output: Weights α_t for the approximated value function $V(\cdot) = \sum_{i=1}^m \alpha_{ti} k(\tilde{\mathbf{x}}_i, \cdot)$

For $t = 1, 2, \dots$ observe transition $(\mathbf{x}_{t-1} \rightarrow \mathbf{x}_t, r_t)$ under (fixed) policy π

1. Sparse approximation for \mathbf{x}_t

Compute $\mathbf{k}_t, \mathbf{h}_t, k_{tt}, h_{tt}$

Obtain \mathbf{p} from $\mathbf{K}_{mm}^{1/2} \mathbf{p} = \mathbf{k}_t$ and \mathbf{a}_t from $\mathbf{K}_{mm}^{T/2} \mathbf{a}_t = \mathbf{p}$

If $k_{tt} - \mathbf{a}_t^T \mathbf{k}_t > \nu$

Add \mathbf{x}_t to the dictionary, $\mathbf{K}_{mm}^{1/2} \leftarrow [\mathbf{K}_{mm}^{1/2}; \mathbf{0}; \mathbf{p}^T, \sqrt{k_{tt} - \mathbf{p}^T \mathbf{p}}]$, **Goto 3.**

Else **goto 2.**

2. Add data point but do not augment basis ($\{\Phi_{t-1,m}^{1/2}, \mathbf{q}_{t-1,m}\} \mapsto \{\Phi_{tm}^{1/2}, \mathbf{q}_{tm}\}$):

Generate Θ_t to lower triangularize the array

$$\begin{bmatrix} \lambda^{1/2} \Phi_{t-1,m}^{1/2} & \mathbf{h}_t \\ \lambda^{1/2} \mathbf{q}_{t-1,m} & r_t \end{bmatrix} \Theta_t = \begin{bmatrix} \Phi_{tm}^{1/2} & \mathbf{0} \\ \mathbf{q}_{tm} & * \end{bmatrix}$$

and read $\Phi_{tm}^{1/2}, \mathbf{q}_{tm}$ from the rhs. Solve $\Phi_{tm}^{T/2} \alpha_t = \mathbf{q}_{tm}$ if necessary.

3. Add data point and augment basis ($\{\Phi_{t-1,m}^{1/2}, \mathbf{q}_{t-1,m}\} \mapsto \{\Phi_{t,m+1}^{1/2}, \mathbf{q}_{t,m+1}\}$):

Compute $\mathbf{c} = \Phi_{t-1,m}^{T/2} \mathbf{a}_t$ and $d = \mathbf{c}^T \mathbf{q}_{t-1,m}$. Then get $\Phi_{tm}^{1/2}, \mathbf{q}_{tm}$ from 2.

Obtain \mathbf{u} from $\Phi_{tm}^{1/2} \mathbf{u} = \lambda \Phi_{t-1,m}^{1/2} \mathbf{c} + h_{tt} \mathbf{h}_t$

Compute $\beta = \sqrt{\lambda \mathbf{c}^T \mathbf{c} + h_{tt}^2 + \Lambda \lambda^t (k_{tt} - \mathbf{a}_t^T \mathbf{k}_t) - \mathbf{u}^T \mathbf{u}}$

Update

$$\Phi_{t,m+1}^{1/2} = \begin{bmatrix} \Phi_{tm}^{1/2} & \mathbf{0} \\ \mathbf{u}^T & \beta \end{bmatrix}, \quad \mathbf{q}_{t,m+1} = \begin{bmatrix} \mathbf{q}_{tm} \\ (\lambda d + h_{tt} r_t - \mathbf{u}^T \mathbf{q}_{tm}) / \beta \end{bmatrix}$$

and solve $\Phi_{t,m+1}^{T/2} \alpha_t = \mathbf{q}_{t,m+1}$ if necessary.

Figure 1. Online policy evalution with sparse LS-SVM at $\mathcal{O}(m^2)$ operations per step, m being the number of basis functions

wished to examine how fast learning occurs when coupling LSTD with different types of function approximation. The transitions were fed successively into the algorithm and the error between approximation and true value function was measured. Function approximators were: (1) our sparsified LS-SVM (Gaussian kernel, $\sigma = 0.02$ with $\nu = 0.1$ and $\nu = 0.01$, $\Lambda = 10^{-3}$), (2) a CMAC (resolution 7x7x7 and 10x10x10), and (3) a RBF-net with fixed centers (spread uniformly on a 12 x 12 grid, $\sigma = 0.02$). The results are shown in Fig. 3 and indicate that our approach is in fact superior as far as generalization is concerned. Also compare the required resources: CMAC uses 343 and 1000 weights, the RBF-net uses 144 weights and our LS-SVM uses 122 and 202 weights.⁶

Optimal control In the second experiment we examine online learning and pit our LSTD-LSSVM against iterative sarsa(λ) with CMAC, which is the standard setup in RL and works well for many problems. To use LSTD for optimal control we employ optimistic policy iteration, and hence include a forgetting factor $\lambda = 0.999$ in the LS-SVM to cope with the non-stationarity of the value function due to the changing policy. The remaining parameters for LS-SVM were unchanged. Since we only consider learning value functions, we supply a generative model for policy improvement. The sarsa(λ) ($\lambda = 0.5$) setup was appropriately modified. As testbed we used the puddle-world from above and the puck-on-hill task from

[9]. Both tasks are episodic with designated terminal states: each episode starts from a random state and proceeds until either a terminal state is reached or the number of steps exceeds a fixed limit of 500 steps. Various performance criteria were considered, see Fig. 2 for the results. In both cases LS-SVM outperforms CMAC in terms of generalization, i.e. number of observed transitions to achieve good performance. Again sparse LS-SVM requires far less weights to achieve this level of accuracy, to obtain satisfactory results with the CMAC in the puddle-world we even needed to double the resolution (20x20x10). Fig. 3 illustrate the general benefit when using LSTD-LSSVM policy evaluation instead of iterative TD: just after running 20 trials and visiting only a small fraction of the state space the approximated value function has the same overall shape as the true value function.

5 Summary and future work

We formulated the problem of estimating the value function from actual trajectories in a regularization framework using least-squares SVM. The key contribution is a QR-based solution algorithm that is fully incremental, i.e. solves the resulting normal equations independent of the number of observed transitions and can hence be applied to *online* RL. In the future we wish to extend the scope of our work to the full RL problem. Among other things, we will address the issue of exploration, stochasticity in transitions, model-free learning through Q-values, and more comprehensive experiments with high-dimensional control tasks. We also aim to explore other criteria for the subset selection mechanism in sparse approximation (e.g. taking

⁶ Unfortunately we cannot directly compare CPU-time, since our algorithm is implemented in C whereas the other two run in Matlab. Some numbers: our algorithm processes 10,000 training examples with 122 weights in ~ 2 secs. Dominating this cost with $\mathcal{O}(m^2)$ is the number of weights m .

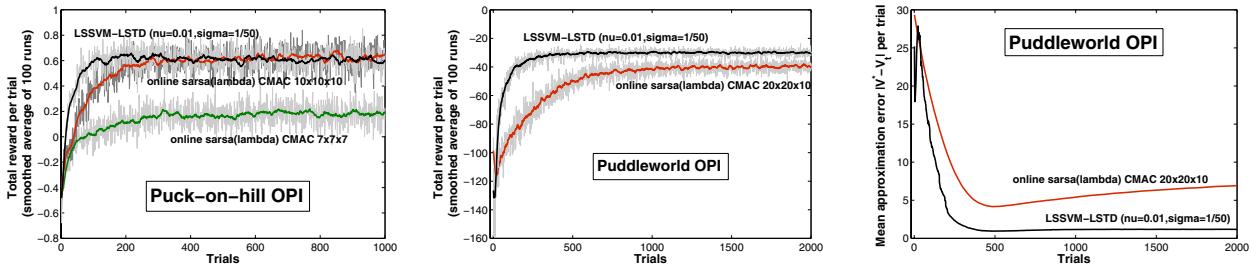


Figure 2. Online learning for optimal control with LS-SVM via optimistic policy iteration (OPI) for two toy problems

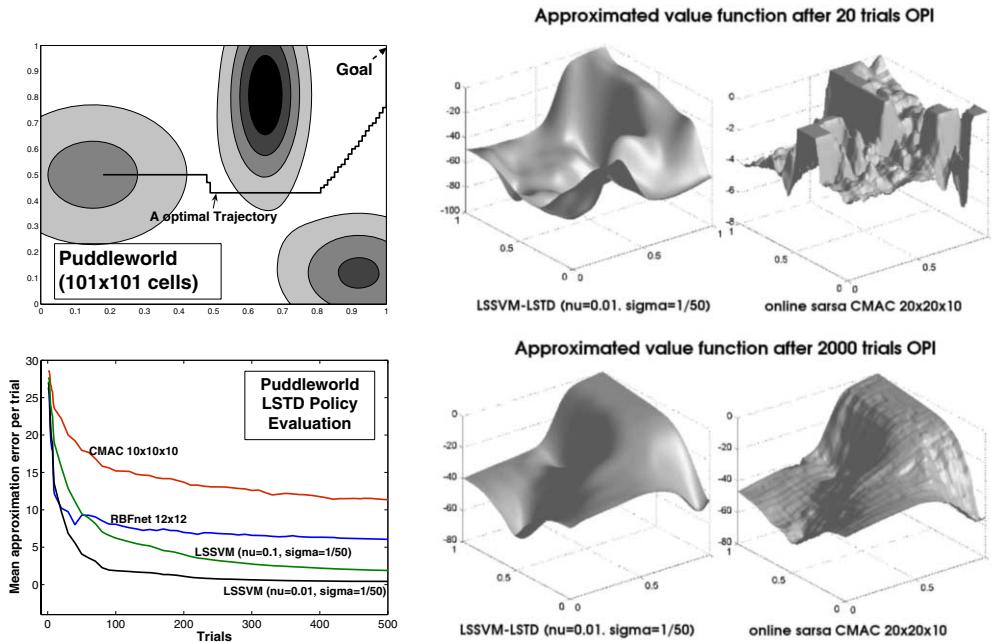


Figure 3. LS-SVM in policy evaluation (left) and OPI (right) for the puddle-world domain

into account the decrease of the error for the final predictor).

We believe that kernel-based methods are very well-suited for RL, since they are data-oriented and thus more flexible than traditional parametric methods: they place parameters where they are needed and when new data arrives and do not waste resources on parts of the state-space that are never visited. Though there is yet not much experimental evidence we believe that this approach could help to tackle high-dimensional control tasks that are yet unsolvable.

REFERENCES

- [1] L. C. Baird, ‘Residual algorithms: Reinforcement learning with function approximation’, in *Proc. of ICML 12*, pp. 30–37, (1995).
- [2] S. J. Bradtko and A. Barto, ‘Linear least-squares algorithms for temporal difference learning’, *Machine Learning*, **22**, 33–57, (1996).
- [3] L. Csató and M. Opper, ‘Sparse representation for Gaussian process models’, in *Advances in NIPS 13*, pp. 444–450, (2001).
- [4] T. Dietterich and X. Wang, ‘Batch value function approximation via support vectors’, in *Advances in NIPS 14*, pp. 1491–1498, (2002).
- [5] Y. Engel, S. Mannor, and R. Meir, ‘Bayes meets Bellman: The Gaussian process approach to temporal difference learning’, in *Proc. of ICML 20*, pp. 154–161, (2003).
- [6] Y. Engel, S. Mannor, and R. Meir, ‘The kernel recursive least squares algorithm’, *IEEE Trans. on Sig. Proc.*, **52**(8), 2275–2285, (2004).
- [7] S. Fine and K. Scheinberg, ‘Efficient SVM training using low-rank kernel representation’, *JMLR*, **2**, 243–264, (2001).
- [8] M. G. Lagoudakis and R. Parr, ‘Least-squares policy iteration’, *JMLR*, **4**, 1107–1149, (2003).
- [9] A. W. Moore and C. G. Atkeson, ‘The parti-game algorithm for variable resolution reinforcement learning in multi-dimensional state-spaces’, *Machine Learning*, **21**(3), 199–233, (1995).
- [10] A. Sayed, *Fundamentals of Adaptive Filtering*, Wiley Interscience, 2003.
- [11] A. J. Smola and P. L. Bartlett, ‘Sparse greedy Gaussian process regression’, in *Advances in NIPS 13*, pp. 619–625, (2001).
- [12] A. J. Smola and B. Schölkopf, ‘Sparse greedy matrix approximation for machine learning’, in *Proc. of ICML 17*, pp. 911–918, (2000).
- [13] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
- [14] C. Williams and M. Seeger, ‘Using the Nyström method to speed up kernel machines’, in *Advances in NIPS 13*, pp. 682–688, (2001).

Argument Based Rule Learning

Martin Možina and Jure Žabkar and Ivan Bratko¹

Abstract. We present a novel approach to machine learning, called ABML (argumentation based ML). This approach combines machine learning from examples with concepts from the field of argumentation. The idea is to provide expert's arguments, or reasons, for some of the learning examples. We require that the theory induced from the examples explains the examples in terms of the given reasons. Thus arguments constrain the combinatorial search among possible hypotheses, and also direct the search towards hypotheses that are more comprehensible in the light of expert's background knowledge. In this paper we implement ABCN2 as an extension of the CN2 rule learning algorithm, and analyze its advantages in comparison with the original CN2 algorithm.

1 INTRODUCTION

A fundamental problem of machine learning is dealing with large spaces of possible hypotheses. Usually, this problem is addressed either by biasing learning towards simpler hypotheses, i.e. applying Occam's razor, or by using experts' domain knowledge for constraining search. Domingos [5] argues that the former approach is valid only if we honestly believe the target phenomenon is simple, and therefore prefers the use of domain knowledge. However, the problem of the latter approach is the difficulty that experts face when they try to articulate their background knowledge. In this paper, we present a novel approach for constraining search, that is based on the expert's knowledge about chosen learning examples rather than on the whole domain. In this approach, the domain expert explains some of the learning examples, and these examples are then, together with other examples, used in the learning algorithm. Explanation of examples is based on argumentation [10], therefore we call these examples argumented examples. The learning from such examples is called argument based machine learning (ABML), introduced by Bratko and Mozina in [1].

In section 2 we describe the motivation and basic ideas for argument based machine learning. In section 3 we develop an actual implementation of argument based machine learning, i.e. argument based CN2 (ABCN2 for short) as an extension of the well-known rule learning algorithm of Clark and Niblett [4]. In section 4, we demonstrate and analyze the advantages of ABCN2 over the original CN2 on a UCI data set.

2 ARGUMENT BASED MACHINE LEARNING: MOTIVATION AND PROBLEM STATEMENT

Argument based machine learning (ABML; outlined by Bratko and Mozina in [1]) is an approach to machine learning that draws on some

concepts of argumentation theory [10]. In this section we will describe the main motivation for this approach, formulate the learning problem for ABML, and contrast it to the usual ML problem.

Usually the problem of learning from examples is stated as:

- Given examples
- Find a theory that is consistent with the examples

We will now extend this problem statement to that of ABML. In ABML, some of the learning examples are augmented by arguments, or reasons, or (partial) justifications according to principles of argumentation [10]. Defeasible argumentation is a branch of artificial intelligence that analyzes processes of reasoning where arguments for and against a certain claim are produced and evaluated. A typical example of such reasoning is a law dispute at court, where plaintiff and defendant give arguments for their opposing claims, and at the end of the process the party with better arguments wins the case.

In our approach to ABML, arguments are used to explain (or argue) why a certain learning example is in the class as given (arguments for = positive arguments) or why it should not be (arguments against = negative arguments). Examples that are accompanied with arguments are called *argumented examples*. Argumented examples are a means for the domain expert to introduce his (partial) domain knowledge to the learning system prior to learning. The practical advantage of providing prior knowledge in the form of arguments about *individual* examples, in comparison with providing more general domain knowledge, is in that it is easier to explain concrete, specific examples than giving general theories. As arguments are reasons for a specific example, the expert does not have to ensure that the arguments are true for the whole domain. It suffices that they are true in the context of the given argumented example.

To illustrate the idea of argumented examples, consider a simple learning problem: learning about credit approval. Each example is a customer's credit application together with the manager's decision about credit approval. Each customer has a name and three attributes: PaysRegularly (with possible values "yes" and "no"), Rich (possible values "yes" and "no") and HairColor ("black", "blond", ...). The class is CreditApproved (with possible values "yes" and "no"). Let there be three learning examples shown in Table 1.

Table 1. Learning examples for credit approval.

Name	PaysRegularly	Rich	HairColor	CreditApproved
Mrs. Brown	no	yes	blond	yes
Mr. Grey	no	no	grey	no
Miss White	yes	no	blond	yes

The expert's argument for disapproving credit to Mr. Grey can be: Mr. Grey did not receive credit because he does not pay regularly. This is a very reasonable argument, however it does not hold for the whole domain. Mrs. Brown, for instance, did receive credit although she doe does not pay regularly.

¹ Faculty of Computer and Information Science, University of Ljubljana, Slovenia, email: martin.mozina@fri.uni-lj.si

With arguments, the learning problem statement changes to:

- Given examples + supporting arguments for some of the examples
- Find a theory that explains the examples using given arguments

Consider again the examples in Table 1 and assume that an expert gave an argument to Mrs. Brown: Mrs. Brown received the credit because she is rich. A rule learning algorithm might induce the following rule:

IF $HairColor = blond$ THEN $CreditApproved = yes$

This rule correctly explains all examples with class value “yes”. However, according to the definition of learning problem with arguments, induced rules should explain the Mrs. Brown using given arguments. Instead of mentioning hair color, an argument based rule learning algorithm would induce:

IF $Rich = yes$ THEN $CreditApproved = yes$

This rule explains Mrs. Brown example using given arguments (Credit was approved because $Rich = yes$).

The main motivation for using arguments in learning lies in two expected advantages:

1. Reasons (arguments) impose constraints over the space of possible hypotheses, thus reducing search complexity
2. An induced hypothesis should make more sense to an expert as it has to be consistent with given arguments.

Regarding advantage 1, by using arguments, the computational complexity associated with search in the hypothesis space can be reduced considerably, and enable faster and more efficient induction of hypotheses. Furthermore, a reduced number of possible hypotheses also decreases chances that the method will overfit. Regarding advantage 2, there are many possible hypotheses that, from the perspective of a machine learning method, explain the given examples sufficiently well. But some of those hypotheses can be incomprehensible to experts. Using arguments should lead to hypotheses that explain given examples in similar terms to those used by the expert, and correspond to the actual justifications.

3 IMPLEMENTATION

In this paper we introduce an algorithm, called ABCN2, for learning rules in argument-based framework for machine learning. ABCN2, which stands for argument based CN2, is an extension of the well-known CN2 rule induction algorithm of Clark and Niblett [4]. We will start this section with a description of argumented examples that are accepted by ABCN2, and continue with a detailed description of differences between the CN2 algorithm and its argument based counterpart ABCN2. At the end of this section we will describe a method that selects examples from data, which, if argumented, would most likely affect the learning of ABCN2.

3.1 Argumented examples

A learning example E in the usual form accepted by CN2 is a pair (A, C) , where A is an attribute-value vector, and C is a class value. In addition to such examples, ABCN2 also accepts argumented examples. An argumented example AE is a triple of the form:

$$AE = (A, C, Arguments)$$

As usual, A is an attribute-value vector and C is a class value. $Arguments$ is a set of arguments Arg_1, \dots, Arg_n , where an argument Arg_i has one of the following forms:

C because $Reasons$

or

C despite $Reasons$

The former specifies a *positive* argument (speaks for the given class value), while the latter specifies a *negative* argument (speaks against the class value). $Reasons$ is a conjunction of reasons r_1, \dots, r_n ,

$$Reasons = r_1 \wedge r_2 \wedge \dots \wedge r_n$$

where each of the reasons r_i can be in one of five possible forms. These forms are (explanations given to forms assume that r_i is a part of a positive argument; for negative arguments, the explanations are exactly the opposite):

- $X = x_i$ means that value x_i of attribute X is the reason why example is in the class as given. This is the only allowed form for discrete attributes.
- $X > x_i$ (or $X \geq x_i$) means that, as the value of attribute X of example is higher (or equal) than x_i , this is the reason for class value.
- $X < x_i$ (or $X \leq x_i$) the opposite to $X > x_i$ ($X \geq x_i$).
- $X >$ (or $X \geq$) “ X is high”, similar to $X > x_i$ ($X \geq x_i$), just that we do not know the threshold value and it has to be found by ABCN2. Such an argument says that the value of X of example is high enough to be in the class as given.
- $X <$ (or $X \leq$); “ X is low”, the opposite of $X >$ ($X \geq$).

For example, the expert’s argument for approving credit to Mrs. Brown (see Table 1) can be: Mrs. Brown received credit because she is rich. A negative argument can be: Mrs. Brown received credit despite she does not pay regularly. The Brown example would in our syntax be written as:

$$((PaysRegularly = no, Rich = yes, HairColor = blond), \\ CreditApproved = yes, \{CreditApproved = yes \text{ because} \\ Rich = yes, CreditApproved = yes \text{ despite} \\ PaysRegularly = no\}).$$

Arguments given to examples additionally constrain rules *covering* this example. Remember that in CN2, rules have the form:

IF $Complex$ THEN $Class$

where $Complex$ is the conjunction of simple conditions, called *selectors*. For the purpose of this paper, a selector simply specifies the value of an attribute, for example $HairColor = blond$ or a threshold on an attribute value, for example $Salary > 5000$. A rule for our credit approval domain can be:

IF $PaysRegularly = no$ AND $HairColor = blond$ THEN
 $CreditApproved = yes$

The condition part of the rule is satisfied by the attribute values of Mrs. Brown example, so we say that this rule *covers* this example.

A rule R is *consistent* with an argument: C because $Reasons$ (or C despite $Reasons$), if for all reasons r_i of $Reasons$ is true that:

1. If the reason r_i is in one of forms: “ $X = x_i$ ” or “ $X > x_i$ ” or “ $X < x_i$ ” (“ $X \geq x_i$ ” or “ $X \leq x_i$ ”), then exactly the same selector needs to be present in the complex of the rule R .
2. If the reason r_i has the form “ $X >$ ” (or “ $X <$ ”, “ $X \leq$ ”, “ $X \geq$ ”), then the complex of the rule R needs to contain a selector “ $X > x_i$ ” (or “ $X < x_i$ ”, “ $X \geq x_i$ ”, “ $X \leq x_i$ ”). The threshold value x_i does not matter for consistency.

With the use of arguments in examples, the definition of a rule *covering* an example needs to be refined. In the standard definition, a rule covers an example if the condition part of the rule is true for this example. In argument based rule learning, this definition is modified to: A rule R AB-covers an argumented example E if:

1. All conditions in R are true for E (same as in CN2),
2. R is consistent with at least one positive argument of E , and
3. R is not consistent with any of negative arguments of E .

As an illustration of the differences between AB-covering and the usual definition of covering, consider again the Brown example with the argument that she received credit because she is rich and despite she does not pay regularly. Now consider four rules:

- R1: IF $HairColor = blond$ THEN $CreditApproved = yes$.
- R2: IF $PaysRegularly = no$ AND $HairColor = blond$
THEN $CreditApproved = yes$
- R3: IF $PaysRegularly = no$ AND $Rich = yes$
THEN $CreditApproved = yes$
- R4: IF $HairColor = blond$ AND $Rich = yes$
THEN $CreditApproved = yes$.

All four rules cover the Brown example and have 100% accuracy on the above data set. However, Rule 1 does not AB-cover the example, because it is not consistent with the positive argument. For the same reason, rule 2 does not AB-cover the Brown example, but this rule fails also because it is consistent with the negative argument ($PaysRegularly = no$). Rule 3 also fails due to the negative argument, although it is consistent with the positive argument. The last example AB-covers the Brown example.

3.2 ABCN2 - Algorithm

The CN2 algorithm [4, 3] consists of a covering algorithm and a search procedure that finds individual rules by performing beam search. The covering algorithm induces a list of rules that cover all the examples in the learning set. Roughly, the covering algorithm starts by finding a rule, then it removes from the set of learning examples those examples that are covered by this rule, and adds the rule to the set of rules. This process is repeated until all the examples are removed.

There are two versions of CN2: one induces ordered list of rules, and the other unordered list of rules. Our algorithm in this paper is based on the second version of CN2. In this case, the covering algorithm consists of two procedures, CN2unordered and CN2ForOneClass. The first procedure iteratively calls CN2ForOneClass for all the classes in the domain, while the second induces rules only for the class given. When removing covered examples, only examples of this class are removed [3]. Essentially, CN2ForOneClass is a covering algorithm that covers the examples of the given class.

3.2.1 ABCN2: Covering algorithm

The first requirement for ABML is that an induced hypothesis explains argumented examples using given arguments. In rule learning, this means that for each argumented example, there needs to be at least one rule in the set of induced rules that AB-covers this example. This is achieved simply by replacing covering in original CN2 with AB-covering.

Replacing the “covers” relation in CN2 with “AB-covers” in ABCN2 ensures that both argumented and non-argumented examples are AB-covered. However, in addition to simply AB-covering all

Algorithm 1 Covering algorithm of ABCN2 algorithm that learns rules from examples ES for given class T

Procedure ABCN2ForOneClass(Examples ES, Class T)

Let RULE_LIST be an empty list.

Let AES be the set of examples that have arguments; $AES \subseteq ES$
Find splits for arguments where threshold not specified (type $X >$, etc.)

Evaluate arguments (as if they were rules) of examples in AES and **sort** examples in AES according to quality of their best argument.

while AES is not empty **do**

 Let AE1 be the first example in AES.

 Let BEST_RULE be *ABFind_best_rule*(ES,AE1,T)

 Add BEST_RULE to RULELIST.

 Remove from AES examples AB-covered by BEST_RULE.

end while

for all RULE in RULE_LIST **do**

 Remove from ES examples AB-covered by RULE.

end for

 Add rules obtained with *CN2ForOneClass*(ES,T) to RULE_LIST

the examples, we would prefer also explaining as many as possible non-argumented examples by arguments given for the argumented examples. Therefore, we propose a change in the covering algorithm, where CN2ForOneClass is changed into ABCN2ForOneClass (see Algorithm 1). The procedure starts by creating an empty list of rules, and makes a separate set AES of argumented examples only. Then it looks for “unfinished” arguments - arguments that have some of the reasons unspecified ($X >$ and $X <$) and finds the best splits for these reasons. Arguments in examples AES are then evaluated², and examples in AES are sorted according to the “best” given argument. Then, the procedure induces a rule, using *ABFind_Best_rule*, to cover the first argumented example. *ABFind_Best_rule* is a modified beam search procedure that accepts examples, an argumented example and a target class, where the resulting rule is guaranteed to AB-cover the given argumented example. This rule is added to the rule set, and the procedure removes from AES argumented examples AB-covered by this rule. The removal of all positive examples is not necessary, as each of the argumented examples differently constrains the search and thus prevents ABCN2 from inducing the same rule again. When all argumented examples are covered, all positive examples AB-covered by rules are removed, and the remaining rules are learned using classical CN2ForOneClass.

3.2.2 ABCN2: Search Procedure

Algorithm 2 shows AB search procedure. The procedure takes a set of examples to learn from, an argumented example that needs to be AB-covered by the induced rule and a target class. In Algorithm 2 the underlined parts emphasize the differences between the original search procedure in CN2 and AB-search procedure:

Initial value of set STAR are positive arguments of example E.

A rule induced from an argumented example must AB-cover this example, therefore it will have to contain the reasons of at least one of positive arguments. The easiest way to ensure this is to start learning from them.

² Evaluation function in CN2 is Laplace’s rule of succession. However, many alternatives were proposed to improve learning. In this document we assume that Laplace is always used unless is specified differently.

Algorithm 2 Algorithm that finds best rule that AB-covers example E from a set of argumented examples. The “quality” of a complex is evaluated by user-defined evaluation function.

Procedure ABFind_Best_Rule(Examples ES, Example E, Class T)

Let the set STAR contain positive arguments of E (written as complexes).

Evaluate complexes in STAR (using quality function).

Let BEST_CPX be the best complex from STAR.

Let SELECTORS be the set of all possible selectors that are TRUE for E

Let ARG_REASON be the set of all reasons in positive arguments of E (union of reasons).

while STAR is not empty **do**

{Specialize all complexes in STAR as follows}

Let NEWSTAR be the set

$\{x \wedge y \mid x \in \text{STAR}, y \in \text{SELECTORS}\}$

Remove from NEWSTAR all complexes that are consistent with any of negative arguments of E.

for every complex C_i in NEWSTAR **do**

if C_i is statistically significant(ES,T) and quality(C_i) > quality(BEST_CPX) **then** {Statistical significance function is inherited from CN2}

replace the current value of BEST_CPX by C_i

end if

end for

Let STAR be best N complexes from NEWSTAR; N is a user-defined size of STAR (usually N=5).

Let ABNEWSTAR be such subset of NEWSTAR,

that complexes in ABNEWSTAR contain only conditions from ARG_REASON.

Let ABSTAR be best N complexes from ABNEWSTAR.

Let STAR be STAR merged with ABSTAR.

end while

return rule: "IF BEST_CPX THEN T."

Specialize with selectors that are satisfied by argumented example.

This ensures the coverage of the seed example by the induced rule.

Remove all complexes that are consistent with negative arguments.

Again, rules must AB-cover argumented example, therefore can not be consistent with any of negative arguments.

Let ABSTAR be best N complexes from ABNEWSTAR Rules that contain only conditions that are present in positive arguments are likely to be consistent with domain knowledge. Thence, these rules are deemed promising, although at the moment they are not among first N best rules according to quality.

It should be noted that argument based ML is not limited to CN2. Various standard ML techniques can be extended in a similar way to their argument-based variants. This document shows only one of possible implementations.

3.3 Identifying critical examples

Giving arguments to all examples is obviously not feasible. We developed a method to automatically find “problematic” examples, that is examples where arguments would be likely to have a significant effect on learning. So the “commentator” of examples (expert who provides arguments) is asked to concentrate on these “problematic” cases. This idea is realised by the following iterative procedure:

1. *Induce a hypothesis without arguments (using classical CN2).*
2. *Find a critical example that needs to be argumented.* This step involves a search for the most problematic example (e.g. outlier) in the learning set. For this task we propose a k-fold cross-validation repeated n times (e.g. n = 4, k = 5), so that each example is tested n times. The example that was misclassified in most of the cases is chosen as the example that needs to be argumented. If there are several such examples, then the algorithm picks a random one.
3. *If problematic example was not found (in step 2), then stop the iteration.*
4. *An expert gives arguments to the selected example.* Two things can happen here: in the preferred case the expert finds argumenting this example easy; in the undesired case, the expert finds this to be hard. The second case may be due to different reasons (deciding attributes are not available, the example is indeed an outlier, or argumenting may be hard in the chosen representation of arguments). Each of these cases can be mended in different ways, which still needs to be explored. In the extreme, this example can simply be discarded from the learning set.
5. *Induce rules on the learning set using ABCN2 and new argumented example.*
6. *Return to step 2.*

4 EXPERIMENTS WITH ABCN2

In this section we will show a simple example to illustrate how ABCN2 works in practice. A larger experiment with ABCN2, an application to law, is described in [6].

4.1 ZOO Data Set

ZOO data set [7] contains descriptions of 101 animals (instances) with 17 attributes: *hair, feathers, eggs, milk, predator, toothed, domestic, backbone, fins, legs, tail, catsize, airborne, aquatic, breathes, venomous, and type*, which is the class attribute. *Type* has seven possible values: *mammal, bird, reptile, fish, amphibian, insect, and other*. The main advantage of this data set is that, using encyclopedia, we can give good arguments to examples. With arguments we expect to increase comprehensibility and classification accuracy of rules.

The set was split to learning set (70%) and test set (30%), evaluation function was m-estimate of probability [2] with $m = 2$, and statistical significance threshold was set to 1.0. The values were obtained with internal-cross validation (only on learn set) to maximize the classification accuracy. Without arguments (classic CN2), the following rules were induced:

IF milk=yes THEN type=Mammal
IF fins=yes AND breathes=no THEN type=Fish
IF feathers=yes THEN type=Bird
IF legs=6 AND breathes=yes THEN type=Insect
IF legs=4 AND hair=no AND predator=yes THEN type=Amphibian
IF legs=0 AND venomous=yes AND catsize=no AND toothed=yes THEN type=Reptile
IF toothed=no AND legs=4 AND hair=no THEN type=Reptile

Considering learning data alone, the induced rules fit perfectly. However, classification accuracy on the test set is only slightly over 90%. Now, according to our method of involving argumented examples (section 3.3), we have to find the most problematic example using the

learning set only. The internal (on learning set only) cross-validation procedure found that the most frequently misclassified example was a reptile, the tortoise. Notice that the rules covering reptiles split reptiles in two subgroups; in the first group are legless, poisonous, small, and toothed reptiles (snakes) and in the other are toothless, with four legs, and hairless reptiles (turtles). A problem with these two rules is that there also exist four-legged reptiles with teeth (crocodile, tuatara, etc. - tuatara was misclassified in the test set). A good argument for tortoise to be a reptile is that it has the backbone and it lays eggs (tortoise is reptile because backbone=yes AND eggs=yes). Using that argument for tortoise in ABCN2, the two rules for reptiles were replaced by the following two rules:

```
IF backbone=yes AND eggs=yes AND aquatic=no AND
feathers=no THEN type=Reptile
IF eggs=no AND breathes=no THEN type=Reptile
```

Naturally, our argument is not a perfect general rule for recognizing reptiles, so it was extended by ABCN2 with attributes *aquatic* and *feathers*. The first attribute separates reptiles from fishes and the second from birds, as both, fishes and birds, have backbone and lay eggs. It is interesting that our argument did not only affect the rule that was learned from this argument, but also the other rule for reptiles.

The next problematic example found was a sea snake (a reptile). A sea snake is an air-breathing snake that lives underwater. According to encyclopedia, a sea snake should be correctly classified with the above rule, however we noticed that, in the data, sea snake is characterized as a non-breathing reptile and that it does not lay eggs. It is obviously a mistake in data and is also the reason for the induction of the second rule.

The next problematic example found in the learn set was a newt. The argument was provided that the newt is an amphibian because it has backbone, lays eggs and is related to water. This resulted in the rule:

```
IF backbone=yes AND aquatic=yes AND eggs=yes AND legs=4
THEN type=Amphibian
```

After adding these arguments to the two examples, ABCN2 induced a perfect set of rules for the ZOO data set. There were no misclassified examples in the test set.

This example clearly illustrates the effectiveness of our method for choosing critical examples (both identified examples were really critical) and shows how arguments can be used to learn concepts that are otherwise very hard to learn. The example also nicely illustrates that it is much easier to give arguments only to an individual example (e.g why tortoise is a reptile), than it would be to articulate general rules that correctly classify the animals. Moreover, the rules learned from arguments are consistent with prior knowledge. Pazzani [9, 8] argued that induced models are more comprehensible if they are consistent with prior knowledge, even if this makes them more complex. Accordingly, we believe that the new set of rules is more comprehensible to an expert.

5 CONCLUSION

In this paper we have described a novel approach to machine learning that draws on some concepts of argumentation theory. We introduced a new type of examples, argumented examples. Arguments are used to constrain the search among possible hypotheses.

We implemented ABCN2, an argument based extension of CN2. We have shown several advantages of argument based approach to learning rules:

- The capture of expert knowledge in the form of arguments for individual examples is easier for the expert than providing general theories.
- The critical examples whose arguments are needed to improve rules, are automatically identified by our method.
- ABCN2 produces more comprehensible rules than CN2, because it uses expert-given arguments to constrain learning.
- On our test data set ZOO, ABCN2 achieved higher classification accuracy than classical CN2. Although this might not be true for all domains, we can expect that arguments will, in general, improve accuracy of hypotheses, if arguments given by experts are better than pure guessing.

There are two main issues about ABML that still need to be researched. The first is a large-scale experimental quantitative assessment of the benefits of argumented examples. For this task we plan to experiment on several larger domains with the help of experts and compare results of ABML to ML. The second issue is to determine how does the quality of arguments given by experts affect learning. This involves experiments with arguments of different quality (for instance expert given vs. random arguments), and a comparison of experts' arguments with slightly changed version of the same arguments. Such experiment could, for instance, discover that experts (for a certain domain) usually give over-specified arguments.

ACKNOWLEDGEMENTS

This work was carried out under the auspices of the European Commission's Information Society Technologies (IST) programme, through Project ASPIC (IST-FP6-002307).

REFERENCES

- [1] Ivan Bratko and Martin Možina, 'Argumentation and machine learning'. In: Deliverable 2.1 for the ASPIC project, 2004.
- [2] B. Cestnik, 'Estimating probabilities: A crucial task in machine learning', in *Proceedings of the Ninth European Conference on Artificial Intelligence*, pp. 147–149, (1990).
- [3] Peter Clark and Robin Boswell, 'Rule induction with CN2: Some recent improvements', in *Machine Learning - Proceeding of the Fifth European Conference (EWSL-91)*, pp. 151–163, Berlin, (1991).
- [4] Peter Clark and Tim Niblett, 'The CN2 induction algorithm', *Machine Learning Journal*, 4(3), 261–283, (1989).
- [5] Pedro Domingos, 'The role of occam's razor in knowledge discovery', *Data Mining and Knowledge Discovery*, 3(4), 409–425, (1999).
- [6] Martin Možina, Jure Žabkar, Trevor Bench-Capon, and Ivan Bratko, 'Argument based machine learning applied to law', in *Argumentation in Artificial Intelligence and Law, Workshop at Tenth International Conference on Artificial Intelligence and Law (ICAIL 2005)*, pp. 87–94, Bologna, (2005). Wolf Legal Publishers.
- [7] P. M. Murphy and D. W. Aha, 'UCI Repository of machine learning databases [<http://www.ics.uci.edu/~mlearn/mlrepository.html>]', Irvine, CA: University of California, Department of Information and Computer Science, 1994.
- [8] M. Pazzani, S. Mani, and W.R. Shankle, 'Beyond concise and colorful: Learning intelligible rules', in *Third International Conference on Knowledge Discovery and Data Mining*, pp. 235–238, Newport Beach, CA, (1997). AAAI Press.
- [9] M. J. Pazzani, 'Influence of prior knowledge on concept acquisition: Experimental and computational results.', *Journal of Experimental Psychology: Learning, Memory and Cognition*, 17, 416–432, (1991).
- [10] Henry Prakken and Gerard Vreeswijk, *Handbook of Philosophical Logic, second edition*, volume 4, chapter Logics for Defeasible Argumentation, 218–319, Kluwer Academic Publishers, Dordrecht etc, 2002.

A Real generalization of discrete AdaBoost

Richard Nock¹ and Frank Nielsen²

Abstract. Scaling discrete AdaBoost to handle real-valued weak hypotheses has often been done under the auspices of convex optimization, but little is generally known from the original boosting model standpoint. We introduce a novel generalization of discrete AdaBoost which departs from this mainstream of algorithms. From the theoretical standpoint, it formally displays the original boosting property; furthermore, it brings interesting computational and numerical improvements that make it significantly easier to handle “as is”. Conceptually speaking, it provides a new and appealing scaling to \mathbb{R} of some well known facts about discrete (ada)boosting. Perhaps the most popular is an iterative weight modification mechanism, according to which examples have their weights decreased iff they receive the right class by the current discrete weak hypothesis. Our generalization to real values makes that decreasing weights affect only the examples on which the hypothesis’ *margin* exceeds its average margin. Thus, while both properties coincide on the discrete case, examples that receive the right class can still be reweighted higher with real-valued weak hypotheses. From the experimental standpoint, our generalization displays the ability to produce low error formulas with particular cumulative margin distributions, and it provides a nice handling of those noisy domains that represent Achilles’ heel for common Adaptive Boosting algorithms.

1 Introduction

In supervised learning, it is hard to exaggerate the importance of *boosting* algorithms. Loosely speaking, a *boosting* algorithm repeatedly trains a moderately accurate learner, gets its *weak* hypotheses, combines them, to finally output a *strong* classifier which boosts the accuracy to arbitrary high levels [9, 10]. (discrete) *Adaboost*, undoubtfully the most popular provable boosting algorithm [5], uses weak hypotheses with outputs restricted to the discrete set of classes that it combines via leveraging coefficients in a linear vote. Strong theoretical issues have motivated the extension of this *discrete* AdaBoost [6] to handle real-valued weak hypotheses as well [6, 11, 15, 18]. Even when only few of them are true generalizations of discrete AdaBoost [11, 18], virtually all share a strong background in convex optimization originally rooted in a “key” to boosting in AdaBoost: a strictly convex exponential loss integrated into a weight update rule for the examples, loss which upperbounds the error and approximates the expected binomial log-likelihood. However, very little is often known for these algorithms from the seminal boosting model standpoint [10, 9, 16], a model which roughly requires conver-

gence to reduced true risk under very weak assumptions (with high probability).

In this paper, we propose a new real AdaBoost, a generalization of discrete AdaBoost that handles arbitrary real-valued weak hypotheses. With respect to former real AdaBoosts, the weight update is fundamentally different as it does not integrate anymore the convex exponential loss; also, the leveraging coefficients for the weak hypotheses differ in the output; finally, these leveraging coefficients are given in closed form and their computation can now easily be delayed until the end of boosting, which is not the case for conventional real AdaBoosts [6, 11, 18]. The major theoretical key feature of this algorithm is that it is a provable boosting algorithm in the original sense. Another point is that it saves computation time with respect to previous generalizations of discrete AdaBoost, that need to approximate the solution of a convex minimization problem at each boosting iteration [11, 18]. From the experimental standpoint, the weight update rule, which does not require anymore the approximation of logarithms or exponentials, is less prone to numerical errors. Finally, it prevents or reduces some numerical instabilities that previous generalizations [11, 18] face when the weak hypotheses reach perfect, or perfectly wrong, classification.

As a matter of fact, it is quite interesting that our algorithm is indeed a generalization of discrete AdaBoost, as when the weak hypotheses have outputs constrained to the set of classes, both algorithms coincide. From this standpoint, our paper also brings a relevant *conceptual* contribution to boosting. Indeed, we give a complete generalization to \mathbb{R} of popular (discrete) boosting properties, and this is sometimes clearly not trivial. For example, discrete AdaBoost is very often presented as an algorithm that reweights lower the examples that have received the right class. Scaled to \mathbb{R} , *this is not true anymore*: lower reweighting occurs **only** for examples on which the classifier’s *margin* exceeds its average margin. Only on the discrete prediction framework do these two properties coincide. Furthermore, this scaling property does not hold for previous real AdaBoosts [6, 11, 15, 18]. For some reasons, our scaling smoothes predictions, and it might explain why experiments clearly display that our algorithm handles noise more efficiently than discrete or real AdaBoosts. Noise handling has soon be described as AdaBoost’s potential main problem [1].

Section 2 presents some definitions, followed by a Section on our generalization of discrete AdaBoost. Section 4 presents and discusses experimental results, and a last Section concludes the paper.

2 Definitions

Our framework is rooted into the original weak/strong learning and boosting frameworks, and Valiant’s PAC model of learnability [5, 10, 19]. We have access to a *domain* \mathcal{X} of observations, which could be $\{0, 1\}^n$, \mathbb{R}^n , etc. . Here, n is the number of description

¹ Université Antilles-Guyane, Département Scientifique Interfacultaire, Campus de Schoelcher, B.P. 7209, 97275 Schoelcher, Martinique, France. E-mail: rnock@martinique.univ-ag.fr

² Sony Computer Science Laboratories Inc., 3-14-13 Higashi Gotanda, Shinagawa-Ku, Tokyo 141-0022, Japan. E-mail: Frank.Nielsen@csl.sony.co.jp

variables. More precisely, we collect *examples*, that is, couples (observation, class) written $(\mathbf{x}, y) \in \mathcal{X} \times \{-1, +1\}$. “+1” is called the *positive* class (or label), and “-1” the *negative* class. In this paper, we deal only with the two-classes case. Well known frameworks exist that allow its extension to multiclass, multilabel frameworks [18]. In this paper, boldfaces such as \mathbf{x} denote n -dimensional vectors, calligraphic faces such as \mathcal{X} denote sets and blackboard faces such as \mathbb{S} denote subsets of \mathbb{R} , the set of real numbers. Unless explicitly stated, sets are enumerated following their lower-case, such as $\{\mathbf{x}_i : i = 1, 2, \dots\}$ for vector sets, and $\{x_i : i = 1, 2, \dots\}$ for other sets (and for vector entries). We make the assumption that examples are sampled independently, following an unknown but fixed distribution D over $\mathcal{X} \times \{-1, +1\}$. Our objective is to induce a classifier or *hypothesis* $H : \mathcal{X} \rightarrow \mathbb{S}$, that matches the best possible the examples drawn according to D .

For this objective, we define a *strong* learner as an algorithm which is given two parameters $0 < \varepsilon, \delta < 1$, samples according to D a set \mathcal{S} of examples, and returns a classifier or *hypothesis* $H : \mathcal{X} \rightarrow \mathbb{S}$ such that with probability $\geq 1 - \delta$, its true risk is bounded as follows:

$$\Pr_{(\mathbf{x}, y) \sim D}[\text{sign}(H(\mathbf{x})) \neq y] = \epsilon_{D, H} \leq \varepsilon. \quad (1)$$

Here, $\text{sign}(a)$ is +1 iff $a \geq 0$, and -1 otherwise. The time complexity of the algorithm is required to be polynomial in relevant parameters, among which $1/\varepsilon, 1/\delta, n$. To be rigorous, the original models [19, 10] also mention dependences on concepts that label the examples. Examples are indeed supposed to be labeled by a so-called *target* concept, which is unknown but fixed. Distribution D is in fact used to retrieve the examples from this target concept, and the time complexity of the algorithm is also required to be polynomial in its size. Hereafter, we shall omit for the sake of clarity this notion of target concept, which is not important for our purpose, since our analysis may also be fit to handle it as well.

A *weak* learner (*WL*) has basically the same constraints, with the notable exception that (1) is only required to hold with $\varepsilon = 1/2 - \gamma$ for some $\gamma > 0$ a constant or inverse polynomial in relevant parameters, and this still has to be verified regardless of D . Since predicting the classes at random, such as with an unbiased coin, would yield $\Pr_{(\mathbf{x}, y) \sim D}[\text{sign}(\text{random}(\mathbf{x})) \neq y] = 1/2, \forall D$, it comes that a weak learner is only required to perform slightly better than random prediction. In the original models, it is even assumed that δ is also an inverse polynomial in relevant parameters, which makes that the constraints on *WL* are somehow the lightest possible from both the statistical and the computational standpoints. The (discrete) *Weak Learning Assumption* (*WLA*) assumes the existence of *WL* [9, 16]. Simple simulation arguments of *WL* [8] allow to show that the weakening on δ is superficial, as we can in fact weak learn with the same arbitrary δ as for strong learning. However, the conditions on ε are dramatically different, and the question of whether weak and strong learning are equivalent models has been a tantalizing problem until the first proof that there exists *boosting* algorithms that strong learn under the *sole* access to *WL* [16], thus proving that these models are indeed equivalent. This first boosting algorithm outputs a large tree-shaped classifier with majority votes at the nodes, each node being built with the help of *WL*. The point is that it is not easy to implement, and it does not yield classifiers that correspond to familiar concept representations.

AdaBoost [5] has pioneered the field of easily implementable boosting algorithms, for which $\mathbb{S} = \{-1, +1\}$. After [5], we refer to it as *discrete* AdaBoost. Basically, AdaBoost uses a weak learner as a subprocedure and an initial distribution \mathbf{w}_1 over \mathcal{S} which is repeatedly skewed towards the hardest to classify examples. After T rounds

Input: sample $\mathcal{S} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, +1\}\}$

$\mathbf{w}_1 \leftarrow \mathbf{u}$;

for $t = 1, 2, \dots, T$ **do**

 Get $(h_t : \mathcal{X} \rightarrow \mathbb{S}) \leftarrow WL(\mathcal{S}, \mathbf{w}_t)$;

 Find $\alpha_t \in \mathbb{R}$;

 Update: $\forall 1 \leq i \leq m$,

$$w_{t+1, i} \leftarrow w_{t, i} \times \exp(-\alpha_t y_i h_t(\mathbf{x}_i)) / Z_t; \quad (2)$$

end

Output: $H_T(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$

Algorithm 1: An abstraction of AdaBoost

of boosting, its output, H_T , is a linear combination of the weak hypotheses. Algorithm 1 gives a useful abstraction of AdaBoost, in which \mathbf{u} is the uniform distribution, Z_t is the normalization coefficient, the elements of \mathcal{S} are enumerated $s_i = (\mathbf{x}_i, y_i)$ and their successive weight vector is noted \mathbf{w}_t , for $t \geq 1$. In discrete AdaBoost, we would have:

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_{\mathbf{w}_t, h_t}}{\epsilon_{\mathbf{w}_t, h_t}}, \quad (3)$$

where “ln” denotes the natural logarithm. The two key steps in AdaBoost are the choice of α_t and the weight update rule. They are strongly related and follow naturally if we seek to minimize the following observed *exponential loss* [6, 18] through the induction of H_T :

$$\epsilon_{\mathbf{w}_1, H_T}^{\text{exp}} = E_{(\mathbf{x}, y) \sim \mathbf{w}_1}(\exp(-y H_T(\mathbf{x}))), \quad (4)$$

with E the mathematical expectation. Since $I[\text{sign}(H_T(\mathbf{x})) \neq y] \leq \exp(-y H_T(\mathbf{x}))$ (with I the indicator function), $\epsilon_{\mathbf{w}_1, H_T} \leq \epsilon_{\mathbf{w}_1, H_T}^{\text{exp}}$, and so minimizing the exponential loss amounts to minimizing the empirical risk as well [6, 11, 15, 17], and it turns out that it brings a boosting algorithm as well [5, 17]. There are other excellent reasons to focus on the exponential loss instead of the empirical risk: it is smooth differentiable and it approximates the binomial log-likelihood [6]. Its stagewise minimization brings both the weight update in (2), and the following choice for α_t :

$$\alpha_t = \arg \min_{\alpha \in \mathbb{R}} E_{(\mathbf{x}, y) \sim \mathbf{w}_t}(\exp(-\alpha y h_t(\mathbf{x}))) \quad (5)$$

$$= \arg \min_{\alpha \in \mathbb{R}} Z_t. \quad (6)$$

One more reason, and not the least, to focus on the exponential loss, is that it brings a stagewise maximization of *margins*. While the empirical risk focuses only on a *binary* classification task (the class assigned is either good or bad), margins scale it to *real* classification, as they integrate both the binary classification task *and* a real magnitude which quantifies a “confidence” in the label given. Large margin classification can bring very fast true risk minimization [17]. Margins justify to scale $\mathbb{S} = \{-1, +1\}$ to an interval such as $[-1, +1]$ [6, 18]; in this case, the sign of the output gives the class predicted. Whenever we still enforce $h_t : \mathcal{X} \rightarrow \{-1, +1\}$, (5) admits a closed-form solution, which is naturally (3), and the boosting algorithm is discrete AdaBoost [6, 5]. Relaxing \mathbb{S} to $[-1, +1]$ yields *real* AdaBoost [6, 11, 18]. Unfortunately, (5) does *not* have a closed-form solution in this case [18]. Iterative techniques exist for its fast approximation [13], but they have to be performed at *each* boosting iteration (which buys overall a significant load increase), and it may be the case that the solution found lies *outside* the boosting regime if the number of approximation steps is too small.

Many other extensions have been proposed to scale $\mathbb{S} = \{-1, +1\}$ up to $[-1, +1]$ or even \mathbb{R} in AdaBoost, but virtually very few of those that do not generalize discrete AdaBoost have been proven so far to be boosting algorithms in the original sense. For space considerations, we focus our comparison only on very few of them that bear at least close similarities with true real generalizations of discrete AdaBoost [6, 11, 15, 18]. These algorithms, that fit to Algorithm 1, mainly differ on the choice of the leveraging coefficient α_t . For example, [6, 15] pick $\alpha_t = 1$, a choice for which the boosting property is not shown to hold.

3 Our Real generalization of AdaBoost

We now give up with the direct minimization of (4), and scale $\mathbb{S} = \{-1, +1\}$ up to \mathbb{R} itself; suppose we replace the weight update (2) and eq. (5) by what follows:

$$w_{t+1,i} \leftarrow w_{t,i} \times \left(\frac{1 - (\mu_t y_i h_t(\mathbf{x}_i)/h_t^*)}{1 - \mu_t^2} \right) . \quad (7)$$

$$\alpha_t = \frac{1}{2h_t^*} \ln \frac{1 + \mu_t}{1 - \mu_t} . \quad (8)$$

Here, we have fixed $h_t^* = \max_{1 \leq i \leq m} |h_t(\mathbf{x}_i)| \in \mathbb{R}$, the maximal value of h_t over \mathcal{S} , and:

$$\mu_t = \frac{1}{h_t^*} \sum_{i=1}^m w_{t,i} y_i h_t(\mathbf{x}_i) \in [-1, +1] \quad (9)$$

the normalized *margin* of h_t over \mathcal{S} . For the sake of clarity, we suppose in the following subsection that $\forall 1 \leq t \leq T, h_t^* < \infty$. Infinite values for h_t^* can be handled in two ways: either we bias/threshold the output of h_t to make it finite [18], or we code it as ∞ , making $\alpha_t = 0$ and $\mu_t = \sum_{i:|h_t(\mathbf{x}_i)|=\infty} w_{t,i} \text{sign}(y_i h_t(\mathbf{x}_i))$. In both cases, \mathbf{w}_{t+1} is a valid distribution and the properties below are kept. Let us call AdaBoost_R our real generalization of discrete AdaBoost.

3.1 Properties

We first show that AdaBoost_R is indeed a generalization of discrete AdaBoost.

Lemma 1 When $\mathbb{S} = \{-1, +1\}$, AdaBoost_R = discrete AdaBoost.

Proof: In this case, we have $h_t^* = 1$ and $\mu_t = 1 - 2\epsilon_{\mathbf{w}_t, h_t}$, which brings that eq. (8) is also $\alpha_t = (1/2) \ln((1 - \epsilon_{\mathbf{w}_t, h_t})/\epsilon_{\mathbf{w}_t, h_t})$, i.e. like in discrete AdaBoost. Our update rule simplifies to:

$$w_{t+1,i} \leftarrow \frac{w_{t,i}(1 - y_i h_t(\mathbf{x}_i) + 2y_i h_t(\mathbf{x}_i)\epsilon_{\mathbf{w}_t, h_t})}{2\epsilon_{\mathbf{w}_t, h_t}(1 - \epsilon_{\mathbf{w}_t, h_t})} ,$$

i.e.:

$$w_{t+1,i} \leftarrow \begin{cases} w_{t,i}/(2(1 - \epsilon_{\mathbf{w}_t, h_t})) & \text{iff } y_i h_t(\mathbf{x}_i) = +1 \\ w_{t,i}/(2\epsilon_{\mathbf{w}_t, h_t}) & \text{iff } y_i h_t(\mathbf{x}_i) = -1 \end{cases} .$$

This is the same expression for the weight update of discrete AdaBoost. \square

Now, we show that AdaBoost_R is a boosting algorithm for arbitrary real-valued weak hypotheses. In fact, we show a little bit more, and for this objective, we define the *margin* of H_T on example (\mathbf{x}, y) as:

$$\begin{aligned} \nu_T((\mathbf{x}, y)) &= \tanh(y H_T(\mathbf{x})/2) \\ &= \frac{\exp(y H_T(\mathbf{x})) - 1}{\exp(y H_T(\mathbf{x})) + 1} \in [-1, +1] . \end{aligned} \quad (10)$$

This definition of margin extends a previous one for discrete AdaBoost [21], and its choice is discussed in Section 3.2. $\forall \theta \in [-1, +1]$ we also define the classifier's "margin error" as the proportion of examples whose margin does not exceed θ :

$$\nu_{\mathbf{u}, H_T, \theta} = \frac{1}{m} \sum_{i=1}^m I[\nu_T((\mathbf{x}_i, y_i)) \leq \theta] . \quad (11)$$

Clearly, $\epsilon_{\mathbf{u}, H_T} = \nu_{\mathbf{u}, H_T, 0}$, and $\nu_{\mathbf{u}, H_T, \theta}$ generalizes $\epsilon_{\mathbf{u}, H_T}$. We now prove a first Theorem on AdaBoost_R.

Theorem 1 $\forall \mathbb{S} \subseteq \mathbb{R}, \forall \theta \in [-1, +1], \text{ after } T \geq 1 \text{ iterations, we have:}$

$$\nu_{\mathbf{u}, H_T, \theta} \leq \left(\frac{1 + \theta}{1 - \theta} \right) \times \exp \left(-\frac{1}{2} \sum_{t=1}^T \mu_t^2 \right) . \quad (12)$$

Proof: We need the following simple Lemma.

Lemma 2 $\forall a \in [-1, 1], \forall b \in [-1, 1]$,

$$1 - ab \geq \sqrt{1 - a^2} \exp \left(-\frac{b}{2} \ln \frac{1 + a}{1 - a} \right) .$$

Proof: The function in the right-hand side is strictly convex in b for $a \neq 0$, and both functions match for $b = \pm 1$ and $a = 0$. Writing the right-hand side $(1 + a)^{(1-b)/2}(1 - a)^{(1+b)/2}$ implies that its limits when $a \rightarrow \pm 1$ are zero. \square

Consider some example $(\mathbf{x}_i, y_i) \in \mathcal{S}$ and some $1 \leq t \leq T$. The way we use Lemma 2 is simple: fix $a = \mu_t$ and $b = y_i h_t(\mathbf{x}_i)/h_t^*$. They satisfy the assumptions of the Lemma, and we obtain:

$$\begin{aligned} 1 - (\mu_t y_i h_t(\mathbf{x}_i)/h_t^*) \\ \geq \sqrt{1 - \mu_t^2} \exp \left(-\frac{y_i h_t(\mathbf{x}_i)}{2h_t^*} \ln \frac{1 + \mu_t}{1 - \mu_t} \right) . \end{aligned} \quad (13)$$

Unraveling the weight update rule, we obtain:

$$\begin{aligned} w_{T+1,i} \times \prod_{t=1}^T (1 - \mu_t^2) \\ = u_i \times \prod_{t=1}^T (1 - (\mu_t y_i h_t(\mathbf{x}_i)/h_t^*)) . \end{aligned} \quad (14)$$

Using T times (13) on the right-hand side of (14) and simplifying yields:

$$(w_{T+1,i}/u_i) \times \prod_{t=1}^T \sqrt{1 - \mu_t^2} \geq \exp(-y_i H_T(\mathbf{x}_i)) . \quad (15)$$

Since for any real q , $I[q \leq 0] \leq \exp(-q)$, we have:

$$\begin{aligned} I[\nu_T((\mathbf{x}_i, y_i)) \leq \theta] &= I \left[y_i H_T(\mathbf{x}_i) - \ln \frac{1 + \theta}{1 - \theta} \leq 0 \right] \\ &\leq \exp \left(-y_i H_T(\mathbf{x}_i) + \ln \frac{1 + \theta}{1 - \theta} \right) \\ &= \left(\frac{1 + \theta}{1 - \theta} \right) \times \exp(-y_i H_T(\mathbf{x}_i)) \\ &\leq \frac{w_{T+1,i}}{u_i} \left(\frac{1 + \theta}{1 - \theta} \right) \times \prod_{t=1}^T \sqrt{1 - \mu_t^2} , \end{aligned}$$

Where the last line uses ineq. (15). There only remains to sum that last ineq. for all examples of \mathcal{S} , and use the fact that $\sqrt{1 - a^2} \leq$

$\exp(-\alpha^2/2), \forall \alpha \in [-1, 1]$. Given that w_{T+1} is a distribution and $u_i = 1/m$, we immediately obtain the statement of the Theorem. \square

Theorem 1 generalizes a well-known convergence Theorem for AdaBoost's empirical risk [18] ($\theta = 0$). This generalization is important, as it says that virtually *any* margin error is subject to the same convergence rate towards zero, and not simply the empirical risk. Thus, more than a single point, it gives also a complete curve $f(\theta)$ upperbounding the empirical margin distribution as given by (11). To prove that AdaBoost_R is a boosting algorithm, we need a WLA that a real-valued WL should satisfy. Its formulation for real-valued hypotheses follows that for the discrete case [9, 10, 18]: basically, it amounts to say that we want h_t to perform significantly different from *random*, a case which can be represented by $\mu_t = 0$. A natural choice is thus fixing the WLA to be ($\forall t \geq 1$):

(real) WLA $|\mu_t| \geq \gamma$, for the same $\gamma > 0$ as in the discrete WLA.

This, in addition, provides us with a generalization of the discrete WLA [9, 16], since we have in this case $\mu_t = 1 - 2\epsilon_{w_t, h_t}$. This brings that either $\epsilon_{w_t, h_t} \leq (1/2) - \gamma/2$, or $\epsilon_{w_t, h_t} \geq (1/2) + \gamma/2$. It has been previously remarked that this second condition, although surprising at first glance since the empirical risk is worse than random, is in fact equivalent to the first from the boosting standpoint, as it "reverses" the polarity of learning: when h_t satisfies the second constraint, $-h_t$ satisfies the first [6].

Now proving that AdaBoost_R is a boosting algorithm amounts first to using Theorem 1 with $\theta = 0$, to obtain under the WLA that after T iterations of AdaBoost_R, we have $\epsilon_{u, H_T} \leq \exp(-T\gamma^2/2)$. Thus, if we run AdaBoost_R for $T = \Omega((1/\gamma^2) \ln m)$, we get an H_T consistent with S . Since T is polynomial in all relevant parameters, classical VC-type bounds on the deviation of the true risk for linear separators [8, 20] immediately bring the following Theorem.

Theorem 2 $\forall S \subseteq \mathbb{R}$, provided WLA holds, AdaBoost_R is a boosting algorithm.

3.2 Discussion

Perhaps one of the most important difference with discrete AdaBoost and its numerous offsprings [5, 6, 4, 18] lies in the fact that they tend to reweight lower the examples that have received the right class. This property is appealing, and has certainly participated to their spread and use. However, when scaling the binary classification problem ($S = \{-1, +1\}$) to \mathbb{R} , for this property to fully integrate the extended framework, it should rely entirely on margins (classes + confidences) instead of just classes. This becomes true with AdaBoost_R: lower reweighting occurs only for examples on which the current weak classifier's margin exceeds its average margin (when $\mu_t > 0$):

$$y_i h_t(\mathbf{x}_i)/h_t^* \geq \mu_t. \quad (16)$$

Thus, there can be examples that receive the right class by h_t , and that have their weights increased. When $\mu_t < 0$, the polarity of boosting (and reweighting) is reversed in the same way as when $\epsilon_{w_t, h_t} > 1/2$ for discrete AdaBoost. Finally, these properties are true generalization of discrete AdaBoost's, as all coincide again on the discrete case.

Margins. It may be helpful to compare our definition of margin with those recently used for real extensions of AdaBoost [6, 17, 18]. Eq. (10) is generally replaced by:

$$\nu_T((\mathbf{x}, y)) = \frac{y H_T(\mathbf{x})}{\sum_{t=1}^T \alpha_t} \in [-1, +1]. \quad (17)$$

Eq. (17) is almost equivalent to the margin of a classifier that we have used for weak hypotheses (μ_t). However, eq. (10), which defines the margin of an *example*, appears to be more convenient for three reasons. The first is statistical, as our definition is in direct relationship with additive logistic models [6]. Suppose we fit H_T to the additive logistic model :

$$H_T(\mathbf{x}) = \ln \frac{p[y = +1|\mathbf{x}]}{p[y = -1|\mathbf{x}]}, \quad (18)$$

with the probabilities $p[.|\mathbf{x}]$ to be estimated while learning. This brings for (10):

$$\nu_T((\mathbf{x}, y)) = y(2p[y = +1|\mathbf{x}] - 1). \quad (19)$$

The quantity $2p[y = +1|\mathbf{x}] - 1 = p[y = +1|\mathbf{x}] - p[y = -1|\mathbf{x}]$ which replaces $H_T(\mathbf{x})/\sum_{t=1}^T \alpha_t = \ln(p[y = +1|\mathbf{x}]/p[y = -1|\mathbf{x}])/\sum_{t=1}^T \alpha_t$ in (17) is the *gentle* logistic approximation of [6], a quantity which is much more stable than the full logistic model itself. The second reason is more technical. Theorem 1 shows that $\nu_{u, H_T, \theta}$ vanishes under the WLA regardless of the value of $\theta \in (-1, 1)$ with margin definition (10). Upperbounds for $\nu_{u, H_T, \theta}$ with eq. (17) are not as easy to read, as all require to vanish that θ be smaller than fluctuating upperbounds that can be $\ll 1$ [17, 18]. It does not seem that it is the boosting algorithm which is responsible, as in our case, using (17) would *not* yield a vanishing $\nu_{u, H_T, \theta}$ when $\theta \geq \max_t |\mu_t|/2$, a situation identical to previous analyses [5, 17, 18]. The third reason is an experimental consequence of the second. Definition (10) makes cumulative margin distributions easier to read, since there is no fluctuating theoretical upperbound < 1 for "boostable" margins.

Computations and numerical stability. A first difference with previous generalizations of discrete AdaBoosts that do not fix *ad hoc* values for α_t [6, 11, 18] is computational. Eq. (5) has no closed form solution in the general case, so they all need to approximate α_t . The problem is convex and single variable, so its approximation is simple, but it needs to be performed at *each* iteration, which buys a significant additional computation time with respect to AdaBoost_R, for which α_t is exact. Approximating has another drawback: if not good enough, the current iteration may lie outside the boosting regime [6, 11, 18].

The extensions of discrete AdaBoost [6, 11, 18] face technical and numerical difficulties to compute α_t when h_t or $-h_t$ reaches consistency, that is, when ϵ_{w_t, h_t} approaches its extremal values, 0 or 1. On the extreme values, there is no finite solution to eq. (5), and thus theoretically no weight update. In our case, the problem does not hold anymore, as the multiplicative update of eq. (7) is never zero nor infinite if we adopt the convention that $0/0 = 1$. Indeed, a numerator equals zero iff all numerators equal zero iff all denominators equal zero. Thus, zeroing any numerator or denominator, which amounts to making either perfect or completely wrong classification for h_t on S , brings *no weight change* in w_{t+1} .

A second, well known technical difficulty for some extensions of discrete AdaBoost [6, 11, 18], occurs when the empirical risk approaches 0 or 1, regions where $|\alpha_t|$ has extremely large regimes. In this case, the numerical approximations to exponentials in eq. (5), with the approximations of α_t , make the computation of the weights very instable. Clearly, large multiplicative coefficients for the weight update are possible for AdaBoost_R. However, instability is less pronounced, since we have split the computation of the leveraging coefficients and that of the weight update, allowing the computation of *all* α_t to be delayed till the end of boosting.

Figure 1. Estimated true risks on 25 domains, comparing discrete AdaBoost [5] (D), AdaBoost_R (U) and the real AdaBoost of [6, 11, 18] (T). For each domain, we put in emphasis the **best** algorithm(s) and the *worst* algorithm(s) out of the three. The last 3 rows count the number of times each algorithm counts respectively among the **best**, second, and *worst*.

Domain	$T = 10$			$T = 50$		
	D	U	T	D	U	T
Balance (2C)	8.73	8.73	9.05	4.44	3.81	4.92
Breast-Wisc	4.51	4.65	4.79	4.22	3.38	4.51
Bupa	34.57	34.57	32.85	30.81	27.71	28.57
Echocardio	31.43	26.43	30.71	30.00	25.71	27.86
Glass2	22.94	18.24	19.41	17.65	17.65	15.88
Hayes Roth (2C)	16.47	24.11	14.71	19.41	14.71	15.88
Heart	18.51	16.67	18.89	19.63	16.67	19.63
Heart-Cleve	23.87	21.29	19.68	17.42	19.35	20.97
Heart-Hungary	19.33	19.33	16.33	21.33	16.33	18.33
Hepatitis	16.47	15.29	17.05	14.71	15.29	18.23
Horse	17.10	16.31	18.16	20.00	16.31	33.68
Labor (2C)	18.33	6.67	11.67	8.33	5.00	8.33
Lung cancer (2C)	27.50	27.50	30.00	25.00	25.00	30.00
LEDeven	9.76	17.07	11.22	10.24	9.51	10.98
LEDeven+17	22.68	21.46	25.60	26.34	25.36	26.10
Monks1	25.18	25.18	16.00	13.39	17.50	1.50
Monks2	34.75	33.93	32.28	34.26	37.70	10.17
Monks3	2.85	3.57	2.14	1.43	1.79	1.79
Parity	45.93	45.19	47.78	46.30	47.78	46.30
Pima (2C)	24.42	24.55	25.32	24.94	24.03	25.71
Vehicle (2C)	26.00	27.17	26.35	25.41	25.29	25.88
Votes	4.78	5.00	5.45	3.86	5.00	5.68
Votes w/o	9.78	8.86	9.78	10.23	10.23	10.45
XD6	20.32	21.64	19.51	15.74	14.92	13.77
Yeast (2C)	28.93	28.73	26.80	26.93	27.33	26.67
#best	7	11	9	7	15	6
#second	9	6	5	9	4	5
#worst	9	8	11	9	6	14

4 Experiments

Experiments were carried out on 25 domains, most of which come from the UCI repository [2]. Domains with more than two classes (indicated 2C) were transformed in a two class problem by grouping all classes but the first into one: sometimes, this brought domains with highly unbalanced classes, thereby complicating even further the learning task. On each domain, we ran discrete AdaBoost [5], AdaBoost_R and the real AdaBoost of [11, 18]. WL is set to a rule (monomial) learner, with fixed maximal rule size (attribute number) r [12, 14]. True risks are estimated using a 10-fold stratified cross validation procedure on the whole data. Since each rule h_t has two possible outputs, y_{ht} has four possible values, and so the analytic solution for α_t of discrete AdaBoost does not apply for real AdaBoost [11, 18]. The true α_t is approximated from (5) using a simple dichotomous search until the relative error does exceed 10^{-6} , using results of [13] to make it faster. Empirically, the execution time for real AdaBoost [6, 11, 18] is on average more than 100 times that of discrete AdaBoost and AdaBoost_R.

General results. We first give some general comments on results that were obtained at early and reasonable stages of boosting, namely after $T = 10$ and $T = 50$ steps of boosting, for a rule learner configured with $r = 2$. Figure 1 summarizes the results obtained. While AdaBoost_R tends to perform the best, interesting patterns emerge from the simulated domains from which we know everything about the concept to approximate. Easy domains such as Monks(1+2) [2] are those on which real AdaBoost performs the best and converges

the fastest. Increasing further T makes that real AdaBoost outstrips even more the two other algorithms. However, as the domain gets complicated, AdaBoost_R becomes the algorithm that beats the other two when T increases. Consider the following domain ordering, from the easiest to the hardest: Monks(1+2) (no noise, no irrelevant attributes), XD6 (10% class noise, one irrelevant attribute), LEDeven (10% attribute noise), LEDeven+17 (LEDeven + 17 irrelevant attributes) [2, 12]. Real AdaBoost beats the other two algorithms on XD6, but another experiment on larger classifiers ($r = 3$; $T = 100$) reveals that AdaBoost_R becomes the winner and approaches Bayes risk with 11.15% error, while discrete and real AdaBoost respectively achieve 11.47% and 12.46% error (statistically worse in that latter case). Winning occurs even sooner on LEDeven ($T = 50$) and LEDeven+17 ($T = 10$). One reason for this phenomenon might be the fact that the reweighting scheme of AdaBoost_R is actually gentler than the others, especially on noisy examples: discrete and real AdaBoost are subject to very large weight update, due to the exponential update rule and the fact that higher reweighting can occur on the sole basis of the binary classification result (good/bad class), even when the classifier has minute confidence on the label it predicts. This cannot happen in our case if the classifier's margin is negative; whenever it is positive, examples that receive the *right* class can still be reweighted higher, counterbalancing higher reweighting for eventual noisy examples. Gentler updating, such as by thresholding, has soon been proposed as a line of research to improve noise handling [4].

Noise handling. In order to shed some more light on noise handling, we have drilled down into the results on domain LEDeven+17 [2]. Its basis is a seven bits problem that describes the ten digits of old pocket calculators. Examples are picked uniformly at random and the ten possible classes and grouped in two: even / odd. Each description variable gets flipped with 10% chances, and seventeen attributes are added, that are irrelevant in the strongest sense [7]. Thus, there is a total of $n = 24$ description variables. Figures 2 and 3 displays cumulative margin distributions that were obtained for couple $(r, T) \in \{2, 4, 6\} \times \{10, 100, 200, 1000\}$. The training margins clearly display that the real AdaBoost of [6, 11, 18] is the fastest to converge to perfect classification, followed by discrete AdaBoost [5], and then by AdaBoost_R (“us”). The testing margins display a completely different pattern: on ten out of twelve results, the estimated true risk is the lowest for AdaBoost_R, and on eight out of ten, the second best is discrete AdaBoost (with a difference best-worst sometimes reaching 6%). Real AdaBoost [6, 11, 18] beats discrete AdaBoost when the classifiers built get complicated: $T \geq 200$ and $r \in \{4, 6\}$. The clear symmetric sigmoid shape (plateau) observed on testing for real AdaBoost on these cases (which is also observable —though less pregnant— for discrete AdaBoost, and almost not observed for AdaBoost_R) indicates that the confidence in classification becomes virtually “infinite” for almost all examples, i.e. for those that receive the right class, and also for those receiving the wrong class. This, we think, indicates a tendency to overfit while trying to model these noisy data. This tendency is clearly less pronounced for AdaBoost_R, and if we look at the margin curves for $\theta \leq 0$, the fact that AdaBoost_R’s curve are almost systematically below both others tends to indicate that AdaBoost_R performs sensibly better than both discrete and real AdaBoosts. This is in accordance with the fact that plotting Bayes rule’s margin curves, following the prediction of the logistic model in (18), seems to always yield a curve shape closer to AdaBoost_R. To see this, Figure 4 plots cumulative margin distributions for Bayes rule’s, for varying amounts of attribute noise in LEDeven+17, ranging from 5% to 50% by steps of 5%. Curves were

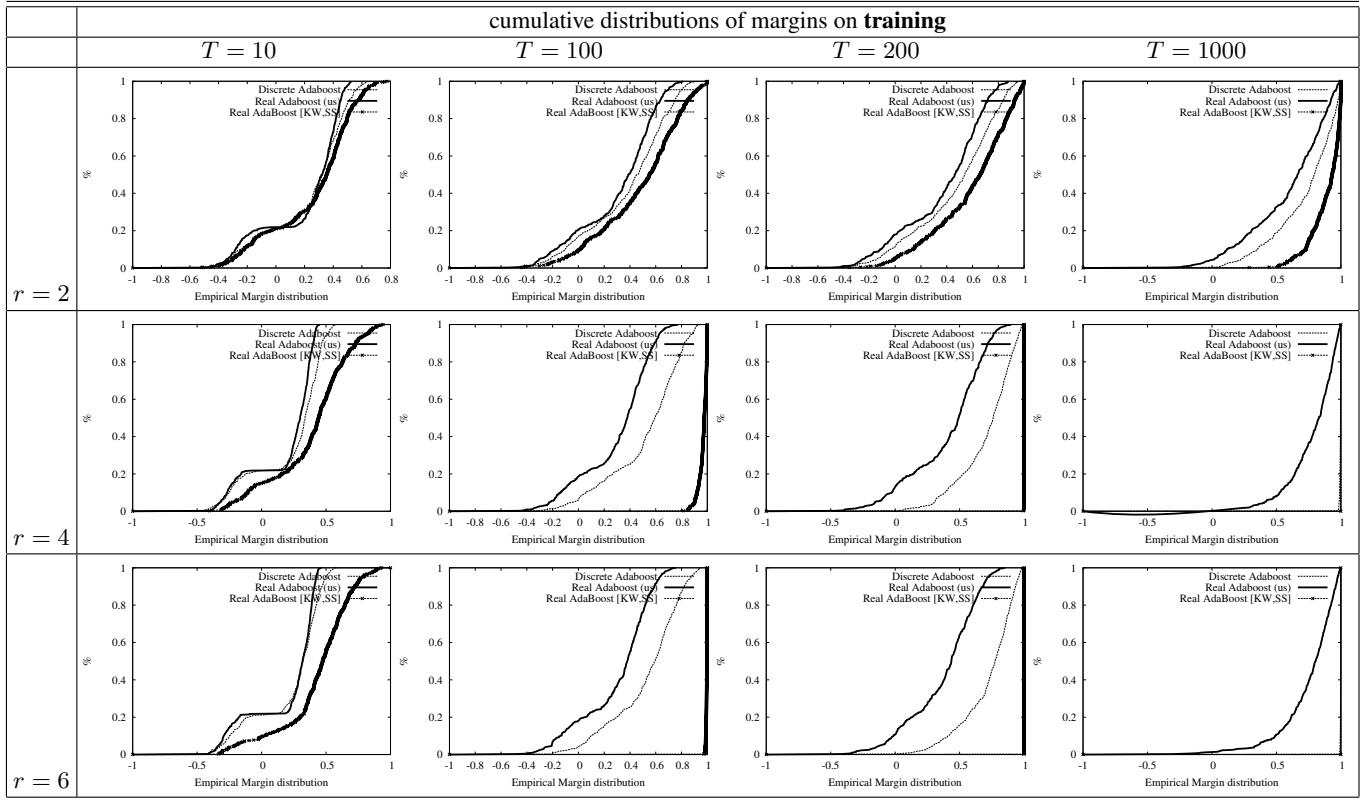


Figure 2. Empirical margin distributions on domain LEDeven+17. An algorithm is as better than another one as its margin distribution is located below the other for negative margin values. The empirical risk of an algorithm is approximately the intersection between its margin distribution and the line $x = 0$.

generated by applying the logistic prediction to random samples containing 300 examples each. It is clear from these curves that the shape that exhibits real AdaBoost as T or r increase, with many examples badly classified with very large confidence, is absent from the logistic prediction. Bayes rule's, along with AdaBoost_R, seem to be more conservative in their predictions. Since the logistic model is fit by discrete and real AdaBoosts [6], this suggests that real AdaBoost, followed by discrete AdaBoost, seem to rapidly overfit the model.

5 Conclusion

In this paper, we have proposed a new generalization of discrete AdaBoost to handle weak hypotheses with real values. Our algorithm, AdaBoost_R, departs from usual generalizations as it does not rely explicitly on the exact minimization of the exponential loss, a loss that upperbounds the empirical risk. While we formally prove that our generalization is a boosting algorithm in the original sense, it provides interesting computational and numerical features with respect to former real extensions of discrete AdaBoost, as well as a generalization of well-known facts about discrete boosting. Among future works, we plan to investigate further the boosting model of learning, and boosting algorithms in a somehow stronger approach. The classical weak and strong learning frameworks are discrete frameworks, as the accuracy of a classifier solely relies on classes. When dealing with real-valued predictions, an accurate framework should take into account both the sign (class), *and* the magnitude (confidence) of the prediction, as a good classifier should basically obtain large confidences with right classes, and not simply right classes. The margin error definition in (11) could be accurate to capture both notions, in

a model that would basically replace $\epsilon_{D,H}$ in (1) by $\nu_{D,H,\theta}$ in (12), with $0 < \theta < 1$ user-fixed, like ε and δ . We believe that strong results are possible: indeed, Theorem 1 shows that all margin errors at fixed $\theta < 1$ vanish with T under the WLA. It should thus be possible to obtain not only classifiers whose true risk is as reduced as desired, *but also* whose confidences are as large as desired. Finally, Theorem 1 does not integrate (the empirical) Bayes risk. A careful integration of this risk, or directly the margin error of Bayes rule, should help to see the way the algorithm behaves on hard problems, and perhaps the way it converges to Bayes rule [3].

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for insightful comments on the paper. R. Nock would like to warmly thank Sony Computer Science Laboratories Inc., Tokyo, for a visiting grant during which part of this work was done.

REFERENCES

- [1] E. Bauer and R. Kohavi, ‘An empirical comparison of voting classification algorithms: Bagging, boosting, and variants’, *Machine Learning*, **36**, 105–139, (1999).
- [2] C. L. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [3] P. Bühlmann and B. Yu, ‘Boosting with the L_2 loss: regression and classification’, *Journal of the American Statistical Association*, **98**, 324–339, (2003).
- [4] C. Domingo and O. Watanabe, ‘MadaBoost: a Modification of AdaBoost’, in *Proc. of the 13th International Conference on Computational Learning Theory*, pp. 180–189, (2000).

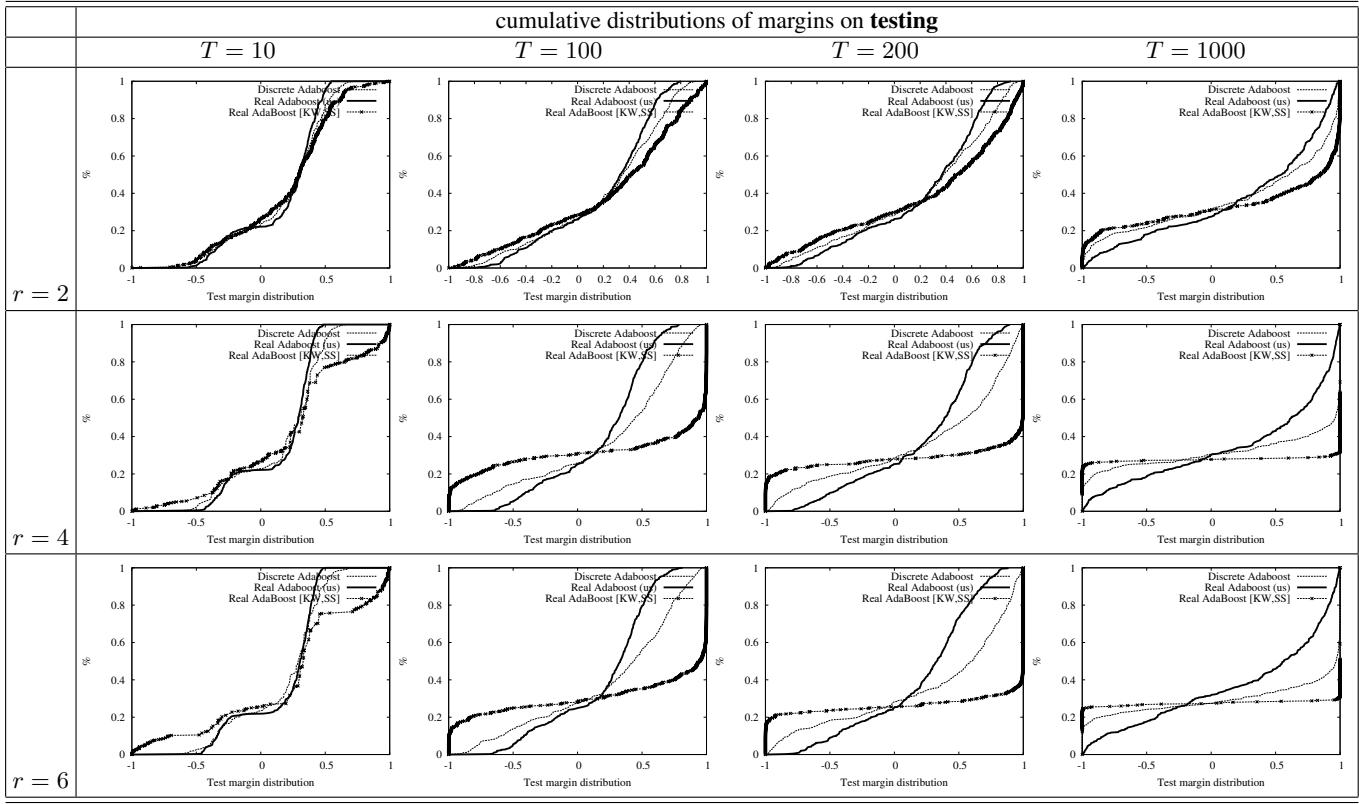


Figure 3. Test margin distributions on domain LEDeven+17. An algorithm is as better than another one as its margin distribution is located below the other for negative margin values. The estimated true risk of an algorithm is approximately the intersection between its margin distribution and the line $x = 0$.

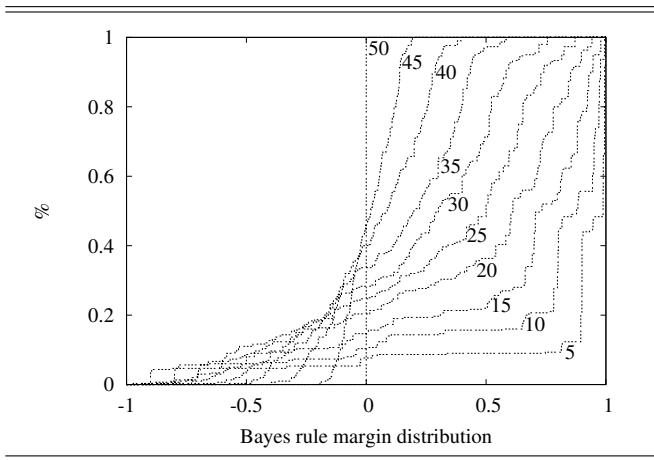


Figure 4. Cumulative distributions of margins for Bayes rule's prediction with the logistic model, on LEDeven+17 with varying amounts of attribute noise, in the set $\{5\%, 10\%, 15\%, 20\%, 25\%, 30\%, 35\%, 40\%, 45\%, 50\%\}$ (see text for details).

- [5] Y. Freund and R. E. Schapire, 'A Decision-Theoretic generalization of on-line learning and an application to Boosting', *Journal of Computer and System Sciences*, **55**, 119–139, (1997).
- [6] J. Friedman, T. Hastie, and R. Tibshirani, 'Additive Logistic Regression : A Statistical View of Boosting', *Annals of Statistics*, **28**, 337–374, (2000).
- [7] G. H. John, R. Kohavi, and K. Pfleger, 'Irrelevant features and the sub-
- set selection problem', in *Proc. of the 11th International Conference on Machine Learning*, pp. 121–129, (1994).
- [8] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*, M.I.T. Press, 1994.
- [9] M.J. Kearns. Thoughts on hypothesis boosting, 1988. ML class project.
- [10] M.J. Kearns and L. Valiant, 'Cryptographic limitations on learning boolean formulae and finite automata', *Proc. of the 21th ACM Symposium on the Theory of Computing*, 433–444, (1989).
- [11] J. Kivinen and M. Warmuth, 'Boosting as entropy projection', in *Proc. of the 12th Int. Conf. on Comp. Learning Theory*, pp. 134–144, (1999).
- [12] R. Nock, 'Inducing interpretable Voting classifiers without trading accuracy for simplicity: theoretical results, approximation algorithms, and experiments', *Journal of Artificial Intelligence Research*, **17**, 137–170, (2002).
- [13] R. Nock and F. Nielsen, 'On weighting clustering', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2006). to appear.
- [14] B. Popescu and J. H. Friedman, 'Predictive learning via rule ensembles', Technical report, Stanford University, (2005).
- [15] G. Ridgeway, 'The state of Boosting', *Computing Science and Statistics*, **31**, 172–181, (1999).
- [16] R. E. Schapire, 'The strength of weak learnability', *Machine Learning*, 197–227, (1990).
- [17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, 'Boosting the margin : a new explanation for the effectiveness of voting methods', *Annals of statistics*, **26**, 1651–1686, (1998).
- [18] R. E. Schapire and Y. Singer, 'Improved boosting algorithms using confidence-rated predictions', *Machine Learning*, **37**, 297–336, (1999).
- [19] L. G. Valiant, 'A theory of the learnable', *Communications of the ACM*, **27**, 1134–1142, (1984).
- [20] V. Vapnik, *Statistical Learning Theory*, John Wiley, 1998.
- [21] I. Witten and E. Frank, *Data Mining: practical Machine Learning tools and techniques with Java implementation*, Morgan Kaufmann, 1999.

Discovery of Entailment Relations from Event Co-Occurrences

Viktor Pekar¹

Abstract. Lexical entailment is knowledge that may prove very useful for a variety of applications that deal with information implicit in a text. In this paper we address the problem of automatic discovery of pairs of verbs related by entailment. A specific challenge in this task is recognition of the direction of entailment in a pair. We model entailment by a number of linguistic cues as to local coherence between clauses. We first investigate the effect of these cues on the quality of the model and then evaluate it against human judgements. Our evaluation reveals that the model is capable of correctly identifying the direction in entailment-related pairs, although their separation from bidirectional pairs proves a more difficult task.

1 INTRODUCTION

The ability of language to express the same content in a variety of forms creates a great challenge for any attempt at deep understanding of natural language text. To deal with this problem at the lexical level, Natural Language Processing (NLP) research has been developing ways to automatically construct resources that describe how various lexical forms can express the same meaning. Many NLP applications can benefit from such a resource: in Question Answering, for example, it can help to make the connection between a question and an answer contained in a corpus, and in Relation Extraction, between an extraction pattern and an actually occurring expression that describes the relevant relation.

Automatic acquisition of paraphrases, expressions that have equivalent meaning, has received significant attention recently. In a paraphrase pair, two expressions (typically, verbs along with their argument structures) denote the same kind of semantic relations between entities. For example, the verbs *buy* and *purchase* both describe the same kind of event, a purchase, as well as the same set of roles different entities play in this event. Occurrence of one verb from this pair in a text signals the same semantic relations between entities as the other verb.

Entailment relations between verbs can also be very helpful in recognizing the same meaning expressed through different lexical forms. However, while in a paraphrase two verbs are semantically equivalent, entailment is a directional, or asymmetric, relation: one verb entails the other, but the converse does not need to hold. For example, *buy* entails *own*, but *own* does not entail *buy* (one can own an object without having bought it). Therefore, to make use of such verb pairs within an NLP application, it is crucial to know the direction of entailment in the pair.

The present paper is concerned with the problem of automatic discovery of entailment relations between verbs from text. In the next section we set the background for the study by describing previous

work. We then define the goal of the study and describe our method for entailment discovery. After that we present results of its experimental evaluation, where we first investigate parameters of the proposed method and then assess the quality of its output against human judgements. Finally, we draw conclusions and outline future work.

2 BACKGROUND

The overall task of entailment acquisition has much in common with that of paraphrase acquisition (e.g., [12], [13], [16]). In both tasks the goal is to discover pairs of related verbs and identify mappings between their argument structures. However, because entailment is an asymmetric relation, similarity between two verbs, an inherently symmetrical relation often used for the discovery of paraphrases, cannot be used for the recognition of the direction of entailment.

To account for the asymmetric character of entailment, a popular approach has been to use lexico-syntactic patterns indicative of entailment. In [4] different types of semantic relations between verbs are discovered using surface patterns (like "*X-ed* by *Y-ing*" for enablement) and assessing the strength of asymmetric relations as mutual information between the two verbs. Similar approaches are adopted by [17] and [8] for the discovery of entailment between verbs and by [6] for causal relations between nouns. Although these techniques achieve high precision, their reliance on surface patterns limits their coverage in that they address only those relations that are regularly made explicit through concrete natural language expressions, and only within sentences.

In tasks involving recognition of relations between entities such as Question Answering and Information Extraction, it is crucial to encode the mapping between the argument structures of two verbs. Pattern-matching often imposes restrictions on the syntactic configurations in which the verbs can appear in the corpus: the patterns employed by [4] and [17] derive pairs of only those verbs that have identical argument structures, and only those that involve a subject and a direct object. The method for discovery of inference rules based on distributional similarity proposed by [12] obtains pairs of verbs with highly varied argument structures, which also do not have to be identical. While the inference rules seem to encompass pairs related by entailment, these pairs are not distinguished from paraphrases and the direction of entailment in such pairs is not recognized.

3 ENTAILMENT RELATIONS

With the view of potential applications, we adopt an operational definition of entailment. We define it to be a semantic relation between verbs where one verb, termed premise p , refers to event e_p and at the same time implies event e_q , typically denoted by the other verb, termed consequence q .

¹ University of Wolverhampton, UK, email: v.pekar@wlv.ac.uk

The goal of entailment acquisition is then to find two linguistic templates each consisting of a verb and slots for its syntactic arguments. In the pair, (1) the verbs are related in accordance with our definition of entailment above, (2) there is a mapping between the slots of the two templates and (3) the direction of entailment is indicated explicitly. For example, in the template pair "*buy(obj:X)* \Rightarrow *belong(subj:X)*" the operator \Rightarrow specifies that the premise *buy* entails the consequence *belong*, and *X* indicates a mapping between the object of *buy* and the subject of *belong*.

As opposed to logical entailment, we do not require that lexical entailment holds in all conceivable contexts and view it as a relation that may be more plausible in some contexts than others. For each verb pair, we therefore wish to assign a score quantifying the likelihood of its satisfying entailment in some random context.

4 APPROACH

The key assumption behind our approach is that the ability of a verb to imply an event typically denoted by a different verb manifests itself in the regular co-occurrence of the two verbs inside locally coherent text, i.e. a set of sentences ordered in a pragmatically relevant manner. This assumption is not arbitrary; as discourse investigations show [1], lexical entailment plays an important role in determining the local structure of discourse. We expect this co-occurrence regularity to be equally characteristic of any pair of verbs related by entailment, regardless of its type and the syntactic behavior of verbs.

The method consists of three major steps. First, it identifies pairs of clauses that are related in the local discourse. We chose a clause rather than a sentence as the unit of analysis, since, as previous work has shown (e.g., [17]), two events related by entailment are often described in the same sentence. From the related clauses, our method then creates templates by extracting pairs of verbs along with relevant syntactic argument positions. Third, the method scores each verb pair in terms of plausibility of entailment by measuring how strongly the premise signals the appearance of the consequence inside the text segment at hand. In the following sections, we describe these steps in more detail.

4.1 Identifying discourse-related clauses

While there is extensive research into ways to instantiate theories of discourse in natural language processing and generation systems (e.g., [3] [10], [9]), in this study we are using a number of surface indicators that might help identify pairs of discourse-related clauses, without trying to build a full discourse representation of text.

Textual proximity. We start by parsing the corpus with a dependency parser, treating every verb with its dependent constituents as a clause. For two clauses to be discourse-related, we require that they appear close to each other in the text. Adjacency of sentences has been previously used to model local coherence [11]. To capture related clauses within larger text fragments, we experiment with windows of text of various sizes around a clause. Since locally related sentences tend to be grouped into paragraphs, we additionally require that the two clauses appear within the same paragraph.

Common event participant. Entity-based theories of discourse such as Centering Theory ([7]) claim that a coherent text segment tends to focus on a specific entity. This intuition has been formalized by [2], who developed an entity-based statistical representation of local discourse and showed its usefulness for assessing coherence between sentences. We also impose this as a criterion for two clauses to be discourse-related: their arguments need to refer to the same

participant, henceforth, **anchor**. We identify the anchor as the same noun lemma appearing as an argument to the verbs in both clauses, considering only subject, object, and prepositional object arguments.

Pronouns as anchors. Studies into local discourse also suggest that salient entities within a discourse segment are frequently referred to with anaphoric expressions, rather than with the same noun or noun phrase. Thus, one can expect that clauses that share the same pronoun are likely to be discourse-related. On the other hand, it seems quite probable that identical pronouns may refer to different entities, especially if the pronouns appear at a large distance from each other and/or there are more than one salient entity in the text segment. We investigate the effect of using identical pronouns as anchors that appear at different distances from each other in the text.

4.2 Constructing templates

Once relevant clauses have been identified, we create pairs of syntactic templates, each consisting of a verb and a label specifying the syntactic role the anchor occupies near the verb. For example, given a pair of clauses *Mr Brown was appointed as a new chief executive*, and *Mr Brown became a chief executive*, the method will extract two pairs of templates: $\{\text{appoint}(\text{obj}:X), \text{become}(\text{subj}:X)\}$ and $\{\text{appoint}(\text{as}:X), \text{become}(\text{comp}:X)\}$

Before templates are constructed, we automatically convert complex sentence parses to simpler, but semantically equivalent ones so as to increase the amount of usable data and reduce noise. We convert passive constructions into active, phrases with past and present participles are turned into predicate structures, coordinated nouns and verbs are separated and treated as separate phrases. If a verb has been used in negative clauses, the particle *not* is appended in front of it in order to distinguish between postulated and negated events.

The output of this step is $V \in P \times Q$, a set of pairs of templates $\{p, q\}$, where $p \in P$ is the premise, consisting of the verb v_p and r_p – the syntactic relation between v_p and the anchor, and $q \in Q$ is the consequence, consisting of the verb v_q and r_q – its syntactic relation to the anchor.

4.3 Measuring asymmetric association

To score the pairs for asymmetric association, we use a procedure similar to the method for learning selectional preferences of verbs proposed by [14].

Each template in a pair is tried as both a premise and a consequence. We quantify the ‘preference’ of the premise p for the consequence q as the contribution of q to the amount of information p contains about its consequences seen in the data. First, we calculate Kullback-Leibler Divergence [5] between two probability distributions, u – the prior distribution of all consequences in the data and w – their posterior distribution given p , thus measuring the information p contains about its consequences:

$$D_p(u||w) = \sum_n u(x) \log \frac{u}{w} \quad (1)$$

where $u(x) = P(q_x|p)$, $w(x) = P(q_x)$, and x ranges over all consequences in the data. Then, the score for template $\{p, q\}$ expressing the association of q with p is calculated as the proportion of q ’s contribution to $D_p(u||w)$:

$$\text{Score}(p, q) = P(q|p) \log \frac{P(q|p)}{P(p)} D_p(u||w)^{-1} \quad (2)$$

In each pair we compare the scores in both directions, taking the direction with the greater score to indicate the most likely premise and consequence and thus the direction of entailment.

In the following two sections we first describe the results of a study of the parametrization of the method and then evaluation of the output generated by its most optimal configuration.

5 PARAMETERS OF THE METHOD

5.1 Task

To explore the effects of different parameter settings for the method, we designed a recognition task similar to that of pseudo-word disambiguation [15]. The task was, given a certain premise, to select its correct consequence out of a pool with several artificially created incorrect alternatives. The advantage of this evaluation design, in comparison with manual evaluation of the direct output of the system, is that it requires minimal human involvement and makes it possible to conduct large-scale experiments.

5.2 Data

The experimental material was created from the BLLIP corpus, a collection of texts from the Wall Street Journal (years 1987-89). We chose 15 transitive verbs with the greatest corpus frequency and used a pilot run of our method to extract 1000 highest-scoring template pairs involving these verbs as a premise. From them, we manually selected 129 template pairs that satisfied entailment.

For each of the 129 template pairs, four false consequences were created. This was done by randomly picking verbs with frequency comparable to that of the verb of the correct consequence. A list of parsed clauses from the BLLIP corpus was consulted to select the most typical syntactic configuration of each of the four false verbs. The resulting five template pairs, presented in a random order, constituted a test item. Figure 1 illustrates such a test item.

```
9.1 * buy(subj X,obj Y) => own(subj X,obj Y)
9.2 buy(subj X,obj Y) => approve(subj X,obj Y)
9.3 buy(subj X,obj Y) => reach(subj X,obj Y)
9.4 buy(subj X,obj Y) => decline(subj X,obj Y)
9.5 buy(subj X,obj Y) => compare(obj X,with Y)
```

Figure 1. An item from the test dataset. The template pair with the correct consequence is marked by an asterisk.

The entailment acquisition method was evaluated on entailment templates acquired from the British National Corpus. Even though the two corpora are quite different in style, we assume that the evaluation allows conclusions to be drawn as to the relative quality of performance of the methods under consideration.

5.3 Recognition algorithm

During evaluation, we tested the ability of the method to select the correct consequence among the five alternatives. Our entailment acquisition method generates association scores for one-slot templates. In order to score the double-slot templates in the evaluation material, we used the following procedure.

Given a double-slot template, we divide it into two single-slot ones such that matching arguments of the two verbs along with the verbs themselves constitute a separate template. For example, "send

(*obj*:X, *to*:Y) \Rightarrow go (*subj*:X, *to*:Y)" will be decomposed into "send (*obj*:X) \Rightarrow go (*subj*:X)" and "send (*to*:Y) \Rightarrow go (*to*:Y)". The scores of these two templates are then looked up in the generated database and averaged. In each test item, the five alternatives are scored in this manner and the one with the highest score was chosen as containing the correct consequence.

The performance was measured in terms of accuracy, i.e. as the ratio of correct choices to the total number of test items. Ties, i.e. cases when the correct consequence was assigned the same score as one or more incorrect ones, contributed to the final accuracy measure proportionate to the number of tying alternatives.

This experimental design corresponds to a random baseline of 0.2, i.e. the expected accuracy when selecting a consequence randomly out of 5 alternatives.

5.4 Results

We first examined the following parameters of the model: the window size, the use of paragraph boundaries, and the effect of the shared anchor on the quality of the model.

5.4.1 Window size and paragraph boundaries

As was mentioned in Section 4.1, a parameter in our model is a threshold on the distance between two clauses, below which the clauses are taken to be discourse-related. We evaluated different values of *k*, the size of the window of clauses around a given clause. We also looked at the effect paragraph boundaries have on the identification of related clauses. Figure 2 shows two curves depicting the accuracy of the method as a function of the window size: the first one describes performance when paragraph boundaries are taken into account (PAR) and the second one when they are ignored (NO_PAR).

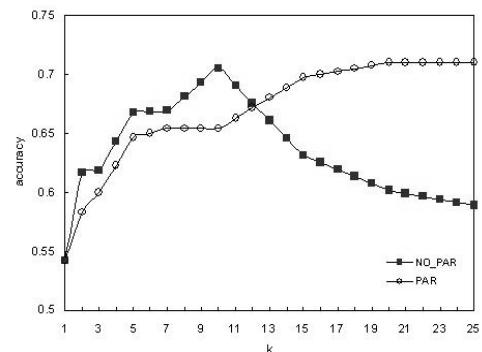


Figure 2. Accuracy of the algorithm as a function of window size, with and without paragraph boundaries used for delineating coherent text.

One can see that both curves rise steeply until *k* = 7, indicating that many entailment pairs are discovered when the two clauses appear close to each other. The rise is the steepest between *k* being 1 and 3, suggesting that entailment relations are most often explicated in clauses appearing very close to each other.

PAR reaches its maximum at *k* = 15, where it levels off. Considering that 88% of paragraphs in BNC contain 15 clauses or less, we take this as an indication that a segment of text where both a premise and its consequence are likely to be found indeed roughly

corresponds to a paragraph. NO_PAR's maximum is at 10, then the accuracy starts to decrease, suggesting that evidence found deeper inside other paragraphs is misleading to our model.

We tested the difference between the maxima of PAR and NO_PAR using the sign test, the non-parametric equivalent of the paired t-test. The test did not reveal any significance in the difference between their accuracies (6-, 7+, 116 ties: $p = 1.000$).

5.4.2 Common anchor

We further examined how the criterion of the common anchor influenced the quality of the model. We compared the configuration that used this criterion (ANCHOR) against the one that did not (NO_ANCHOR), i.e. considering only co-occurrence of verbs concatenated with specific syntactic role labels. Additionally, we included into the experiment a model that looked at plain verbs co-occurring inside a context window (PLAIN). Figure 3 compares the performance of these three models (paragraph boundaries were taken into account in all of them).

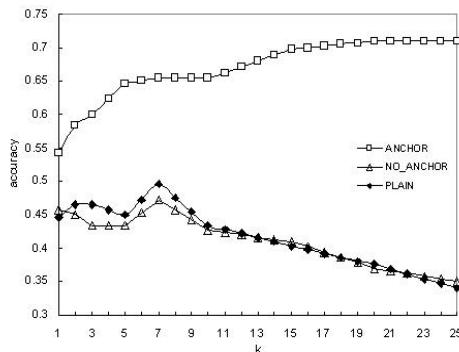


Figure 3. The effect of the common anchor on the accuracy of the method.

Compared with ANCHOR, the other two models achieve considerably worse accuracy scores. The differences between the maximum of ANCHOR and those of the other two models are significant according to the sign test (ANCHOR vs NO_ANCHOR: 44+, 8-, 77 ties: $p < 0.001$; ANCHOR vs PLAIN: 44+, 10-, 75 ties: $p < 0.001$). Their maxima are also reached sooner (at the window of 7) and thereafter their performance quickly degrades. This indicates that the common anchor criterion is very useful, especially for locating related clauses at larger distances in the text.

5.4.3 Pronouns as anchors

Figure 4 compares the performance of the method with and without pronouns being used as anchors for the discovery of template pairs. We see that the two curves are very similar. The use of pronouns offers improvement at $k < 5$, though it is quite small (max. 4% at $k = 3$). With the further increase of k , the performance curve generally stays below the one when pronouns are not used as anchors.

The observed effect of pronouns is consistent with the expectation that identical pronouns occurring close to each other typically refer to the same entities, thus signifying relatedness between clauses. At greater distances, they tend to refer to different entities.

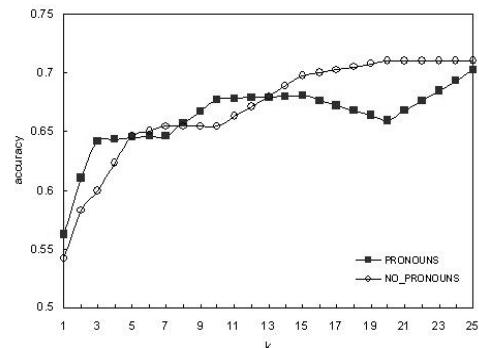


Figure 4. The effect of pronouns used as anchors on the accuracy of the method.

6 HUMAN EVALUATION

6.1 Design

In order to directly evaluate the quality of entailment patterns discovered by our model, we elicited judgements from humans. We randomly selected 300 pairs from the top 1000 scoring pairs generated by the optimized configuration of the model from the BNC data, that was determined in the previous set of experiments (i.e., using a window of 20 verbs, paragraph boundaries and the common anchor criterion; pronouns were not used as anchors).

To examine the ability of the model to capture the direction of entailment relations, from each of the 300 pairs we created a similar one by swapping the premise and the consequence, so that the resulting pair was identical to the original one except that its direction was the opposite. This produced an evaluation set with 600 verb pairs.

Presented with the pairs in a random order, five paid subjects were asked to mark in each pair whether or not the verb on the left implied the one on the right. They were specifically instructed to pay attention to the order of the templates in a pair and mark highly probable, but not necessarily certain implications.

The obtained judgements were then analyzed to arrange the initial 300 pairs into four groups:

- CORRECT, where entailment held in the predicted direction, i.e. the premise implied the consequence, as assigned by the method;
- OPPOSITE, where entailment was found in the direction opposite to the predicted one, i.e. the intended premise was in fact a consequence and vice versa;
- BIDIRECTIONAL, where entailment held in both directions, i.e. the two templates were paraphrases of each other; and
- NO_RELATION, where entailment existed in neither direction.

Table 1 shows 20 randomly selected verb pairs that were present in the CORRECT category with at least three subjects.

6.2 Results

Table 2 shows the mean number of pairs for each of the four groups of verb pairs, standard error as well its proportion in the entire sample.

These results indicate that the model discovers a greater number of pairs with correctly identified direction of entailment in them, than pairs where entailment holds in the opposite direction. A two-tailed

sell X to Y \Rightarrow Y own X	ask X for Y \Rightarrow X have Y
sell X to Y \Rightarrow Y accept X	X eat Y \Rightarrow X get Y
X grant Y \Rightarrow X have Y	X exceed Y \Rightarrow X reach Y
appoint X as Y \Rightarrow X become Y	X play in Y \Rightarrow involve X in Y
X win Y \Rightarrow X hold Y	X prevent Y \Rightarrow X reduce Y
X achieve Y \Rightarrow X have Y	identify X in Y \Rightarrow Y have X
X explain Y \Rightarrow X understand Y	send X to Y \Rightarrow X go to Y
entitle X to Y \Rightarrow X claim Y	X pick_up Y \Rightarrow X take Y
X publish Y \Rightarrow X produce Y	X live in Y \Rightarrow X come to Y
X win Y \Rightarrow X get Y	X wear Y \Rightarrow X have Y

Table 1. A sample of system-generated verb pairs evaluated by the majority of subjects as satisfying entailment.

	Mean	Std. error	%
CORRECT	48.8	3.35	16.26
OPPOSITE	26.4	4.7	8.8
BIDIRECTIONAL	71.4	15.96	23.8
NO_RELATION	153.4	22.26	51.13

Table 2. Four types of system-discovered pairs created on the basis of judges' responses.

paired t-test between CORRECT and OPPOSITE shows the difference is significant ($p = 0.001$). However, the model has difficulty distinguishing between pairs with the correct direction and those with bidirectional relations. The difference between the means is not significant ($p > 0.1$). The evaluation also shows that in about half of the pairs output by the model, neither verb was perceived as implying its counterpart.

6.3 Error analysis

Manual examination of pairs appearing in groups other than CORRECT revealed the following most common types of errors.

Pairs with bidirectional relations made up a large proportion of the system output. They included pairs that would normally be classified as paraphrases (e.g., X cause $Y - X$ lead to Y , X help $Y - X$ encourage Y , X develop $Y - X$ promote Y) and conversives (e.g., X receive $Y - send Y to X$, X buy $Y - sell Y to X$, X buy $Y - X$ pay for Y). This shows that these lexical relations can frequently be found in discourse-related clauses. If the co-occurrence rate for such pairs is high compared with the frequency of one of the verbs, the association score for such pairs turns out to be high. A possible way to filter out such pairs might be to additionally score pairs using a method for discovery of symmetrical relations, such as one that measures distributional similarity between the templates ([12]).

Among pairs in the NO_RELATION group, there is a considerable number of antonyms, such as X reject $Y - X$ approve Y , X lose $Y - X$ win Y , and X reduce $Y - X$ increase Y . Manual inspection suggests there are around 30 antonymous pairs (10%) in the 300 pairs used for the experiment. Because antonymy is a symmetrical relation, the number of output antonyms might also be reduced by a method that discovers symmetrically related verb pairs.

A considerable number of pairs that seem to satisfy entailment ended up in the NO_RELATION category, because of ambiguity of verbs and the insufficiency of information as to the context in which they should be interpreted. For example, X obtain $Y - X$ apply for Y found to be unrelated by 3 out of 5 subjects. However, considering this pair in the context of some legal procedure, entailment does seem

to hold. Similar examples of this kind are X encourage $Y - X$ increase Y , X receive $Y - X$ claim Y , X maintain $Y - X$ build Y . It should also be noted that some pairs that appear in the CORRECT category would normally be judged to be paraphrases in some contexts, such as X regard as $Y - X$ call Y , X approve $Y - X$ pass Y .

7 CONCLUSION

We have proposed a method for automatic discovery of entailment relations from text, a task that is of potential benefit for many AI applications. The central assumption behind the method is that entailment relations manifest themselves in the regular co-occurrence of two verbs inside locally coherent text. We examined the effect on the quality of our model produced by a number of surface cues as to discourse relatedness between clauses, finding that the distance between clauses, paragraph boundaries, and shared arguments of the verbs are good indicators of entailment relations. A direct evaluation of entailment pairs found by the proposed method demonstrated that it provides a promising approach for discovery of the asymmetric relations of entailment.

REFERENCES

- [1] N. Asher and A. Lascarides, *Logics of Conversation*, Cambridge University Press, 2003.
- [2] R. Barzilay and M. Lapata, ‘Modeling local coherence: an entity-based approach’, in *Proceedings of ACL’05*, pp. 141–148, (2005).
- [3] S. Brennan, M. Friedman, and C. Pollard, ‘A centering approach to pronouns’, in *Proceedings of ACL’87*, pp. 155–162, (1987).
- [4] T. Chklovski and P. Pantel, ‘VERBOCEAN: Mining the web for fine-grained semantic verb relations’, in *In Proceedings of Empirical Methods in Natural Language Processing (EMNLP’04)*, (2004).
- [5] T.M. Cover. and J.A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.
- [6] R. Girju, ‘Automatic detection of causal relations for question answering’, in *Proceedings of the ACL’03 Workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond"*, (2003).
- [7] B. Grosz, A. Joshi, and S. Weinstein, ‘Centering : a framework for modeling the local coherence of discourse’, *Computational Linguistics*, **21**(2), 203–225, (1995).
- [8] T. Inui, K. Inui, and Y. Matsumoto, ‘What kinds and amounts of causal knowledge can be acquired from text by using connective markers as clues?’, in *Proceedings of the 6th International Conference on Discovery Science*, pp. 180–193, (2003).
- [9] N. Karamanis, M. Poesio, C. Mellish, and J. Oberlander, ‘Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus’, in *Proceedings of ACL’04*, pp. 391–398, (2004).
- [10] R. Kibble and R. Power, ‘Optimizing referential coherence in text generation’, *Computational Linguistics*, **30**(4), 401–416, (2004).
- [11] M. Lapata, ‘Probabilistic text structuring: experiments with sentence ordering’, in *Proceedings of ACL’03*, pp. 545–552, (2003).
- [12] D. Lin and P. Pantel, ‘Discovery of inference rules for question answering’, *Natural Language Engineering*, **7**(4), 343–360, (2001).
- [13] B. Pang, K. Knight, and D. Marcu, ‘Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences’, in *Proceedings of HLT-NAACL’2003*, (2003).
- [14] P. Resnik, *Selection and Information: A Class-Based Approach to Lexical Relationships*, Ph.D. dissertation, University of Pennsylvania, 1993.
- [15] H. Schütze, ‘Context space’, in *Fall Symposium on Probabilistic Approaches to Natural Language*, pp. 113–120, (1992).
- [16] I. Szpektor, H. Taney, I. Dagan, and B. Coppola, ‘Scaling web-based acquisition of entailment relations’, in *Proceedings of Empirical Methods in Natural Language Processing (EMNLP’04)*, (2004).
- [17] K. Torisawa, *Questions and Answers: Theoretical and Applied Perspectives*, chapter An unsupervised learning method for commonsensical inference rules on events, University of Utrecht, 2003.

Efficient Knowledge Acquisition for Extracting Temporal Relations

Son Bao Pham and Achim Hoffmann¹

Abstract. Machine learning approaches in natural language processing often require a large annotated corpus. We present a complementary approach that utilizes expert knowledge to overcome the scarceness of annotated data. In our framework KAFTIE, the expert could easily create a large number of rules in a systematic manner without the need of a knowledge engineer. Using KAFTIE, a knowledge base was built based on a small data set that outperforms machine learning algorithms trained on a much bigger data set for the task of recognizing temporal relations. Furthermore, our knowledge acquisition approach could be used in synergy with machine learning algorithms to both increase the performance of the machine learning algorithms and to reduce the expert's knowledge acquisition effort.

1 Introduction

Recent years have seen a growing interest in Natural Language Processing applications. Machine learning became the method of choice for building domain specific text analysis systems. Statistical NLP proved very popular as shallow approaches to language understanding appear more promising than previously thought. However, limitations of what those techniques can achieve are on the horizon: machine learning approaches are limited by the available tagged data, in particular, where supervised learning is involved.

There has been an increasing demand for temporal analysis. A number of approaches have addressed temporal processing: identification and normalization of time expressions [6], time stamping of event clauses [4], temporally ordering of events [5], recognizing time-event relations in TimeML [1]. At a higher level, these temporal expressions and their relations are essential for the task of reasoning about time, for example, to find contradictory information [3]. In this emerging domain, there is a clear lack of a large annotated corpus to build machine learning classifiers for detecting temporal relations.

We present our incremental knowledge acquisition approach that utilizes expert's knowledge (almost any competent speaker of the relevant language) to overcome the scarceness of annotated data. We demonstrate the effectiveness of our approach as we have quickly developed a large knowledge base (KB) based on a small data set that performs better than machine learning (ML) approaches trained on a much bigger data set on the task of recognizing temporal relations.

We further show how ML algorithms can be used in synergy with our approach when annotated data is available to both increase the performance of the ML algorithms and reduce the knowledge acquisition effort. This work is based on [7, 8].

Our knowledge acquisition framework builds on the idea of Ripple Down Rules (RDR) [2] which allows one to add rules to the given KB

incrementally while automatically ensuring consistency with previously seen cases i.e the KB's performance is continuously improved. Let us consider simple examples in the task of assigning a positive or negative class to sentences. Given the following sentence:

Our approach has a good performance.

the user could assign the positive class to the sentence due to the word *good* by creating the following rule:

if the text contains "good" then assign positive class (1)

The following sentence:

The approach's performance is not good.

will be assigned with the positive class by rule (1). Suppose the user disagrees as the presence of the word *not* negates the effect of having the word *good*. He or she can in turn create an exception to rule (1):

if the text contains "not" then assign negative class (2)

This rule is potentially applicable only in the context where rule (1)'s condition is satisfied i.e. it is an exception of rule (1). Effectively, the above two rules are equivalent to:

**if the text contains "good" then assign positive class
except if the text contains "not" then assign negative class**

For the sentence:

The work does not follow traditional approaches.

rule (1)'s condition is not satisfied. Rule (2) will therefore not be considered either. However, the sentence

This approach is good and does not suffer from existing problems.

will be labeled incorrectly with the negative class by rule (2). This is probably not the intention of the user when creating the exception rule (2) as the word *not* does not modify the meaning of the word *good*. The user could add another exception rule to refine rule (2):

if "not" appears after "good" then assign positive class (3)

The above examples use a very simple pseudo rule language to illustrate how the knowledge acquisition process works. The actual rule language description used is presented in detail in section 3.1.

In section 2 we present a short overview of RDR. This is followed by a description of KAFTIE in section 3. Section 4 reports experimental results on recognizing temporal relations which demonstrates the success of our approach. Section 5 discusses how ML could be utilized in KAFTIE. The conclusions follow in section 6.

¹ School of Computer Science and Engineering, University of New South Wales, Australia, emails: {sonp,achim}@cse.unsw.edu.au

2 Knowledge Acquisition Methodology

In this section we present the basic idea of Ripple-Down Rules (RDR) [2] which inspired our approach. RDR allows one to add rules to a knowledge base incrementally without the need of a knowledge engineer. A new rule is only created when the KB performs unsatisfactorily on a given case. The rule represents an explanation for why the conclusion should be different from the KB's conclusion on the case at hand.

A *Single Classification Ripple Down Rules* (SCRDR) tree is a binary tree with two distinct types of edges. These edges are typically called *except* and *if-not* edges. See figure 1. Associated with each node in a tree is a *rule*. A rule has the form: *if* α *then* β where α is called the *condition* and β the *conclusion*.

Cases in SCRDR are evaluated by passing a case (an object to be classified) to the root of the tree. At any node in the tree, if the condition of a node N 's rule is satisfied by the case, the case is passed on to the exception child of N using the *except* link if it exists. Otherwise, the case is passed on to the N 's *if-not* child. The conclusion given by this process is the conclusion from the last node in the RDR tree which *fired* (satisfied by the case). To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default* node. We will show how a more complex *default* rule can improve the knowledge acquisition process in section 5.

A new node is added to an SCRDR tree when the evaluation process returns the wrong conclusion. The new node is attached to the last node in the evaluation path of the given case with the *except* link if the last node is the *fired* rule. Otherwise, it is attached with the *if-not* link.

With the structure of RDR tree, it is possible that redundancy is present as one rule could appear in multiple places. Reorganizing RDR structure was reported in [11]. However, this is not crucial as the structure of the KB is transparent to the users.

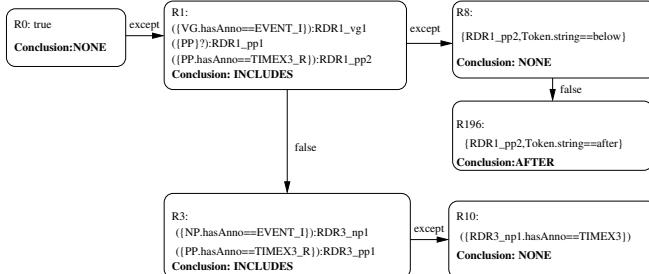


Figure 1. An extract of a KB for recognizing TLINK between an EVENT and a TIMEX3. Note that SCRDR is different from a decision tree: rules in internal nodes can be used to give conclusions to input cases.

3 Our KAFTIE framework

We use SCRDR for building knowledge bases in the KAFTIE (Knowledge Acquisition Framework for Text classification and Information Extraction) framework. Some levels of abstraction in the rule's condition is desirable to make the rule expressive enough in generalizing to unseen cases. To realize this, we use the idea of annotations where phrases that have similar roles (belong to the same concept) are deemed to belong to the same annotation type. Annotations contain the annotation type, the character locations of the beginning and ending position of the annotated text in the document, and a list of feature value pairs.

3.1 Rule description

A rule is composed of a condition part and a conclusion part. A condition is a regular expression pattern over annotations. It can also post new annotations over matched phrases of the pattern's sub-components. The following is an example of a pattern which posts an annotation over the matched phrase:

$\{\{\text{Noun}\}\{\text{VG.type==FVG}\}\{\text{Noun}\}\}:\text{MATCH}$

This pattern would match phrases starting with a Noun annotation followed by a VG, which must have feature *type* equal to FVG, followed by another Noun annotation. When applying this pattern on a piece of text, MATCH annotations would be posted over phrases that match this pattern. As annotations have feature value pairs, we can impose constraints on annotations in the pattern by requiring that a feature of an annotation must have a particular value.

The rule's conclusion contains the classification of the input text. In the task of recognizing temporal relations between a pair of temporal expressions (event or time), the conclusion is either the relation type or NONE.

3.2 Annotations and Features

Built-in annotations: As our rules use patterns over annotations, the decision on what annotations and their corresponding features should be are important for the expressiveness of rules. The following annotations and features make patterns expressive enough to capture all rules we want to specify for various tasks.

We have Token annotations that cover every token with *string* feature holding the actual string, *category* feature holding the POS and *lemma* feature holding the token's lemma form.

As a result of the Shallow Parser module, which is a cascade of finite state transducers, we have several forms of noun phrase annotations ranging from simple to complex noun phrases, e.g. NP (simple noun phrase), NPL (list of NPs) etc. There are also VG (verb groups) annotations with *type*, *voice*, *headVerbPos*, *headVerbString* etc. features and other annotations e.g. PP (prepositional phrase), SUB (subject), OBJ (object). The Shallow Parser used in this work is not geared towards the task.

Every rule that has a non-empty pattern would post at least one annotation covering the entire matched phrase. Because rules in our knowledge base are stored in an exception structure, we want to be able to identify which annotations are posted by which rule. To facilitate that, we number every rule and enforce that all annotations posted by rule number x have the prefix RDR x _. Therefore, if a rule is an exception of rule number x , it could use all annotations with the prefix RDR x _ in its condition pattern.

Apart from the requirement that an annotation's feature must have a particular value, we define extra annotation constraints: *hasAnno*, *hasString*, *underAnno*, *endsWithAnno*. For example, *hasAnno* requires that the text covered by the annotation must contain the specified annotation:

$\text{NP.hasAnno == TIMEX3}$

only matches NP annotations that have a TIMEX3 annotation covering their substring. This is used, for example, to differentiate a TIMEX3 in a noun group from a TIMEX3 in a verb group.

Custom annotations: Users could form new named lexicons during the knowledge acquisition process. The system would then post a corresponding annotation over every word in those lexicons. Doing this makes the effort of generalizing the rule quite easy and keeps the KB compact.

3.3 The Knowledge Acquisition Process in KAFTIE

The knowledge acquisition process goes through a number of iterations. The user gives a text segment (e.g. a sentence) as input to the KB. The conclusion (e.g. classification) is suggested by the KB together with the *fired* rule R that gives the conclusion. If it is not the default rule, annotations posted by the rule R are also shown (see section 4.3) to help the user decide whether the conclusion is satisfactory.

If the user does not agree with the KB's performance, there are two options: adding an exception rule to rule R or modifying rule R if possible. In either case, user's decision will be checked for consistency with the current KB before it gets committed to the KB. To create a new exception rule, the user only has to consider why the current case should be given a different conclusion from rule R . This effort does not depend on the knowledge base size. The level of generality of the new rule is not crucial as this process can be supported by the work reported in [9].

4 Experiments

We build a knowledge base using KAFTIE to recognize TLINK relations between an EVENT and a TIMEX3 in the TimeBank corpus. According to the specification of TimeML [10], TIMEX3 marks up explicit temporal expressions such as times, dates, durations etc. EVENT covers elements in a text describing situations that occur or happen while TLINK is a temporal link representing a relation between an event and a time or between two events.

4.1 The TimeBank corpus

The TimeBank corpus is marked up for temporal expressions, events and basic temporal relations based on the specification of TimeML. Currently, the TimeBank corpus has 186 documents.

Excluding TIMEX3s in document's meta-data (doc creation time), the majority of TLINKs are between EVENTS and TIMEX3s within the same sentence. Hence, in all experiments, we focus on recognizing intra-sentential temporal relations.

The TimeBank annotation guidelines suggest distinctions among TLINK types between two EVENTS but do not explicitly specify how those types are different when it comes to relations between an EVENT and a TIMEX3. In fact, some of the TLINKs types between an EVENT and a TIMEX3 are hard to distinguish and a number of cases of inconsistency are observed in the corpus. We group similar types together: BEFORE and IBEFORE are merged, AFTER and IAFTER are merged and the rest is grouped into INCLUDES.

4.2 KAFTIE for extracting Temporal Relations

For the task of extracting EVENT-TIMEX3 temporal relations, we consider all pairs between an EVENT and a TIMEX3 in the same sentence and build a knowledge base to recognize their relations. The sentence containing the pair is used as the input to the knowledge base. As there could be more than one EVENT or TIMEX3 in the same sentence, we change the EVENT and the TIMEX3 annotations in focus to EVENT_I (instance) and TIMEX3_R (related_to) annotations respectively. This enables our rule's pattern to uniquely refer to the pair's arguments. For each pair, the KB's conclusion is the type of its temporal relation if there exists a relation between the pair's arguments or NONE otherwise.

Apart from this, there was no prior design required to adapt KAFTIE to the task of recognizing temporal relations.

4.3 How to build a Knowledge Base in KAFTIE

The following examples are taken from the actual knowledge base discussed in section 4.4 and shown in figure 1. Suppose we start with an empty KB for recognizing temporal relations between an EVENT and a TIMEX3 within the same sentence. I.e. we would start with only a default rule that always produces a *NONE* conclusion. When the following sentence is encountered:

```
Imports of the types of watches [VG [EVENT_I totaled EVENT_I] VG]
[PP about $37.3 million PP] [PP in [NP [TIMEX3_R 1988 TIMEX3_R]
NP] PP], ...
```

our empty KB would use the default rule to suggest the relation between EVENT_I and TIMEX3_R is *NONE* i.e. no relation exists. This can be corrected by adding the following rule to the KB:

Rule 1:
 $((\{VG.\text{hasAnno} == \text{EVENT_I}\}):RDR1_vg1$
 $(\{\text{PP } ?\}:RDR1_pp1$
 $(\{\text{PP}.\text{hasAnno} == \text{TIMEX3_R}\}):RDR1_pp2$
 $):RDR1_$
Conclusion: INCLUDES

Each of the component in the rule's pattern is automatically assigned a tag which will effectively post a new annotation over the matched token strings of the component if the pattern matches a text. New tags of rule i always start with $RDRi_$. This rule would match phrases starting with a VG annotation which covers the EVENT_I annotation, followed by an optional PP annotation followed by a PP annotation covering the TIMEX3_R annotation. When the sentence containing the pair EVENT_I-TIMEX3_R is matched by this rule, the pair is deemed to be of INCLUDES relation type. Once matched, new annotations RDR1_vg1, RDR1_pp1 and RDR1_pp2 will be posted over the first VG, the first PP and the last PP in the pattern respectively. This is to enable exception rules to refer to the results of previously matched rules. Notice here that the first PP component is specified optional. It could be that the expert already *anticipates* future cases and makes the current rule more general. Alternatively, experts always have a choice of modifying existing rule to cover new cases.² When we encounter this pair:

```
The company's shares [RDR1_vg1 [VG are [EVENT_I wallowing
EVENT_I] far VG] RDR1_vg1] [RDR1_pp2 [PP below [NP their
[TIMEX3_R 52-week TIMEX3_R] NP] PP] RDR1_pp2] high....
```

Rule **R1** fires suggesting that the pair has INCLUDES relation with the newly posted annotation highlighted in bold face. This is incorrect as there is no relation. The following exception rule is added to fix the misclassification:

Rule 8:³
 $(\{RDR1_pp2.\text{Token.string} == \text{below}\}):RDR8_$
Conclusion: NONE

This rule says that if the second PP matched by Rule 1 (RDR1_pp2) starts with a token string *below* then there is no relation between the pair. Notice that the sentence could have different PP annotations. As each rule posts unique annotations over the matched phrases, we can unambiguously refer to relevant annotations.

² Automatic recommendation on which existing rules and how to modify to cover new cases is reported in [9].

³ We only select some rules to show as examples, hence indices of rules are not consecutive

4.4 Experimental results

Out of 186 documents in the TimeBank corpus with approximately 1350 intra-sentential TLINK instances, we randomly took half of that as training data and keep the remaining half for testing. Using our KAFTIE framework, we built a knowledge base of 229 rules to recognize relations between an EVENT and a TIMEX3 in the same sentence using the training data.⁴ The knowledge base uses NONE as its default conclusion. In other words, by default and initially when the KB is empty, all EVENT-TIMEX3 pairs are deemed not to have any TLINK relations.

The overall results are shown in table 1 for two settings: *3 types* (BEFORE, INCLUDES and AFTER see section 4) and *without typing* - collapsing all TLINK types into one type, effectively detecting if the pair has a TLINK relation regardless of the type. While the accuracy reflects performance on the test data across all types including NONE, the F-measure is based on only TLINKs types, i.e. excluding NONE.⁵ On the *without typing* setting, the built knowledge base achieved an F-measure of more than 75% and an accuracy of 86.7%.

Methods	3 types		w/o typing	
	F-m	Acc.	F-m	Acc.
KAFTIE	71.3%	86.1%	75.4%	86.7%
J48 (5-folds)	62.3%	78.7%	66.4%	81.2%
SMO (5-folds)	61.4%	77.4%	63.1%	78.7%

Table 1. Results on recognizing TLINKs for 3 types and *without typing* settings.

For comparison with standard machine learning approaches, we use Weka's J48 and SMO [12] as implementations for C4.5 and support vector machine algorithms respectively. To adapt machine learning algorithms to the task of extracting relations, we define the following feature representation which is capable of capturing the relation arguments (EVENTs and TIMEX3s) and the surrounding context. We break up the sentence containing the EVENT-TIMEX3 pair into five segments namely: spans of the two arguments (EVENT and TIMEX3), the span between the two arguments and spans to the left/right of the left/right arguments. From each segment, we use token strings, token lemmas, parts-of-speeches, bigrams of parts-of-speeches and all annotations from the Shallow Parser as features.

J48 and SMO are run using 5-fold cross validation. As the result could vary depending on the seed used for the cross validation, we report results averaged over 100 runs with different random seeds. As can be seen in table 1, the knowledge base built using our framework significantly outperforms standard J48 and SMO. In fact, our knowledge base with the initial 60 rules (as the result of seeing roughly 60 misclassified TLINK pairs) already outperforms J48 and SMO (see figure 2).

Figure 2 shows the performance of our knowledge base on the test data as rules are incrementally added. Given the upwards trend of the graph as more rules are added, it is plausible that our KB would get to even higher performance had we used 80% of the available data for building the KB.

⁴ To evaluate the TLINK recognition task alone, we use the EVENT and TIMEX3 annotations in the TimeBank corpus.

⁵ The NONE type occurs approximately 2.5 times more often than the TLINK types.

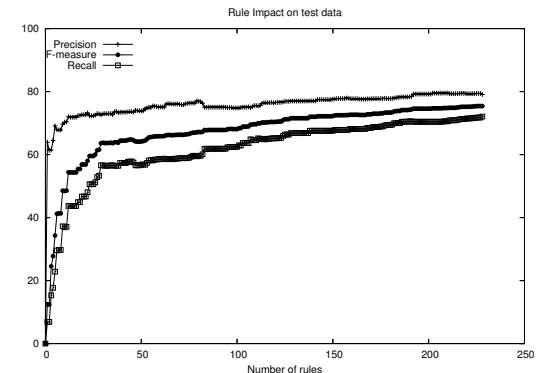


Figure 2. Impact of incrementally adding rules to the KB.

5 Combining ML with KA

In this section, we investigate how machine learning could be used in synergy with our knowledge acquisition approach to both improve the machine learning algorithms' performance and reduce the required knowledge acquisition effort.

A knowledge acquisition (KA) session corresponds to the creation of a new rule as a result of the KB performing incorrectly on a particular case. Notice that the KB's default rule is quite simple. It always returns the default conclusion which is *NONE* for the task of recognizing temporal relations. We conjecture that if we start with a better default rule then the number of knowledge acquisition sessions can be reduced. One way of doing it is to take some training data to train a classifier as the default rule.

We carry out the following experiment: We focus on the task of recognizing temporal relation in the *without typing* setting and use the exact training and test data from the previous section to build and test a KB respectively. The difference is that we now split the training data into two parts i.e. *ML data* and *KB data*. The *ML data* is used to train a classifier as the default rule in a KB while the *KB data* is used to add exception rules to the KB.

Instead of having real experts involved in the process of creating exception rules, we simulate it by consulting the KB built from the previous section, called *oracleKB*.⁶ For each example in the *KB data* that is misclassified by the current KB, we use the *fired* rule for the example from the *oracleKB* i.e. the rule that the *oracleKB* would use on the example.

Table 2 shows the results of using J48 and SMO which are trained on varying portions of the training data. Specifically, it shows the f-measure of the classifier alone, the KB with the classifier as the default rule on the test data as well as the number of rules in the KB. The number of rules in the KB reflects the number of KA sessions required for building the KB. All figures are averaged over 20 different runs using different seeds on splitting the training data into *ML data* and *KB data*.

In all situations, machine learning algorithms can always be augmented with more rules using our knowledge acquisition approach to achieve a significantly better performance, see figure 3.

As we increase the percentage of *ML data*, the number of KA sessions required gets smaller. Ideally, we want to minimize the number of KA sessions while maximizing the f-measure. When the *ML data* is empty (0% of the training data), we effectively rebuild a KB in

⁶ Every rule in this KB is made independent of other rules by incorporating all of its context e.g. the rule it is an exception of.

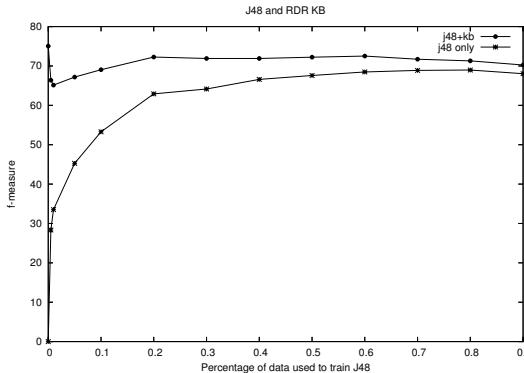


Figure 3. F-measures of J48 and J48 with KB.

the same fashion as in the previous section with different orders of examples. The f-measure of 75.1% suggests that the performance of our approach is independent of the order of examples presented to the experts.

As can be seen from table 2, the f-measures of the KBs with SMO or J48 as the default rule follow the same trend. It first degrades as we start giving some data to the classifier and improves as more data is used to train the classifier. After certain thresholds, the f-measures start degrading again as we get less data for experts to add exception rules while the classifiers do not improve their performance.

Depending on the task and the classifier used, we can choose an appropriate amount of data to train a classifier as the default rule in order to substantially reduce the number of KA sessions involved while still achieve a reasonable performance. For example with J48 the best performance is at 72.5% when we use 60% of the training data for training J48 and the rest to add exceptions rules. This reflects a 4% improvement in f-measure to the raw machine learning classifier. Compared to a fully manual approach (using 0% data for the classifier), we achieve a 60% reduction in the number KA sessions.

As the *oracleKB* is built with the assumption that the default rule always returns *NONE*, all of the exception rules at level 1 (exception rules of the default rule) are rules covering *INCLUDES* instances. Even though rules at level 2 cover *NONE* instances, they have limited scopes because they were created as exceptions to level 1 rules. When the default rule gives an incorrect *INCLUDES* conclusion, it is likely that we would not be able to consult the *oracleKB* for an exception rule to fix this error.⁷ It therefore suggests that if we use real experts, we could achieve a better result.

6 Discussion and Conclusions

We have shown that with our unconventional Knowledge Acquisition approach using KAFTIE, we could quickly build a knowledge base that performs better than the classifiers built by existing machine learning approaches while requiring much less data.

It took 7 minutes on average to create one rule, independent on the current size of the KB. This includes the time needed to read the sentence to understand why there is a certain relation between the pair of an EVENT and a TIMEX3 as well as the time required to test the new rule before committing it to the KB. If the users have to classify the pair's relation from scratch, when we do not have annotated data,

%	j48	j48 +kb	#rules	smo	smo +kb	#rules
0%	0	75.1	173	0	75.1	173
0.5%	28.4	66.4	146	27	68.6	151
1%	33.5	65.2	143	27.3	71.6	158
5%	45.3	67.2	138	54.3	71.9	142
10%	53.3	69	131	61.7	72.4	133
30%	64.2	71.9	103	65.6	72.7	108
50%	67.6	72.2	78	66.6	71.8	84
60%	68.5	72.5	65	67	71.7	70
90%	68	70.3	22	66	68.5	25

Table 2. Results of using j48/SMO as the default rule for KBs averaged over 20 random seeds.

then the actual time spent on creating a rule would be much less, as understanding the sentence takes most of the time.

We further demonstrated how ML algorithms could be used together with our approach to reduce the knowledge acquisition effort. This also results in performance improvement to the ML algorithms.

In future work, we will investigate the application of our framework KAFTIE to tasks in a different language other than English.

REFERENCES

- [1] Branimir Boguraev and Rie Kubota Ando, ‘TimeML-compliant text analysis for temporal reasoning’, in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 997–1003, UK, (2005).
- [2] Paul Compton and R. Jansen, ‘A philosophical basis for knowledge acquisition’, *Knowledge Acquisition*, 2, 241–257, (1990).
- [3] Richard Fikes, Jessica Jenkins, and Gleb Frank, ‘JTP: A system architecture and component library for hybrid reasoning.’, in *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando Florida, USA, (2003).
- [4] Elena Filatova and Eduard Hovy, ‘Assigning time-stamps to event-clauses’, in *Proceedings of the 10th Conference of the EACL*, Toulouse, France, (2001).
- [5] Inderjeet Mani, Barry Schiffmann, and Jiaping Zhang, ‘Inferring temporal ordering of events in news’, in *Proceedings of the NAACL*, Edmonton, Canada, (2003).
- [6] Inderjeet Mani and George Wilson, ‘Robust temporal processing of news’, in *Proceedings of the 38th annual meetings of the ACL*, Hong Kong, (2000).
- [7] Son Bao Pham and Achim Hoffmann, ‘Incremental knowledge acquisition for building sophisticated information extraction systems with KAFTIE’, in *5th International Conference on Practical Aspects of Knowledge Management*, pp. 292–306, Vienna, Austria, (2004). Springer-Verlag.
- [8] Son Bao Pham and Achim Hoffmann, ‘Incremental knowledge acquisition for extracting temporal relation.’, in *Proceedings of 2nd IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE05)*, pp. 354–359, Wuhan, China, (2005). IEEE.
- [9] Son Bao Pham and Achim Hoffmann, ‘Intelligent support for building knowledge bases for natural language processing (in print).’, in *Perspectives of Intelligent Systems’ Assistance*, ed., Roland Kaschek, Idea Group Inc., (2006).
- [10] J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R. Gaizauskas, A. Setzer, G. Katz, and D. Radov, ‘TimeML: Robust specification of event and temporal expressions in text.’, in *AAAI Spring Symposium on New Directions in Question Answering.*, Standford, CA, (2003).
- [11] Hendra Suryanto, *Learning and Discovery in Incremental Knowledge Acquisition.*, Ph.D. dissertation, University of New South Wales, Australia, 2005.
- [12] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, 2000.

⁷ A better simulation is to build another *oracleKB* with the default rule always returning the *INCLUDES* conclusion.

Efficient Learning from Massive Spatial-Temporal Data through Selective Support Vector Propagation

Yilian Qin and Zoran Obradovic¹

Abstract. In the proposed approach, learning from large spatial-temporal data streams is addressed using the sequential training of support vector machines (SVM) on a series of smaller spatial data subsets collected over shorter periods. A set of representatives are selected from support vectors corresponding to an SVM trained with data of a limited spatial-temporal coverage. These representatives are merged with newly arrived data also corresponding to a limited space-time segment. A new SVM is learned using both sources. Relying on selected representatives instead of propagating all support vectors to the next iteration allows efficient learning of semi-global SVMs in a non-stationary series consisting of correlated spatial datasets. The proposed method is evaluated on a challenging geoinformatics problem of aerosol retrieval from Terra satellite based Multi-angle Imaging Spectro Radiometer instrument. Regional features were discovered that allowed spatial partitioning of continental US to several semi-global regions. Developed semi-global SVM models were reused for efficient estimation of aerosol optical depth from radiances with a high level of accuracy on data cycles spanning several months. The obtained results provide evidence that SVMs trained as proposed have an extended spatial and temporal range of applicability as compared to SVM models trained on samples collected over shorter periods. In addition, the computational cost of training a semi-global SVM with selective support vector propagation (SSVP) was much lower than when training a global model using spatial observations from the entire period.

1. INTRODUCTION

With recent advances in remote sensing technologies, scientists collect a much larger volume of spatial-temporal observations than previously. For example, since 2000, the Multi-angle Imaging SpectroRadiometer (MISR) instruments onboard Terra satellite of NASA's Earth observing system have been used to observe solar radiation through nine cameras each in four spectral bands with 1.1 km spatial resolution and global coverage of Earth in about 9 days [1]. Such a resolution results in quick accumulation of terabytes of high dimensional spatial-temporal observation data. The objective of learning nonlinear relationships from such non-stationary data can in principle be addressed by artificial neural networks (ANN) and support vector machines (SVM) [2]. While theoretically such prediction models are more

accurate with large training datasets, the computational cost of learning from high volume data prohibits the training and retraining of global models using the entire historical data.

In a recent study [3], a neural network ensemble procedure was developed to learn ANN models from massive datasets. In this project, neural networks of low complexity were learned from small datasets first, and the component networks were incrementally combined into an ensemble while progressively increasing the size of the training dataset. This allowed for a regulation of the computational costs of developing nonlinear regression models from large datasets through an automated identification of an appropriate number of component networks, their complexity and a sample size for training the component.

For learning nonlinear relationships in high dimensional data, SVM are frequently preferred when both stability and high accuracy are required [4]-[6]. However, the training and prediction time of SVM increases with the increased number of support vectors. Therefore, both training and prediction time may be prohibitively long when learning with an SVM for massive spatial-temporal datasets that typically require a large number of support vectors. Approaches considered to address this situation include reducing the size of datasets [7], improving the performance of SVM through incorporating known invariance of the problems [8], approximating the decision surface and reformulation of the training problem that yields the same decision surface using a smaller number of basis functions [9][10], using a proximal SVM [11], and exploiting partial/merge k-mean method with parallel computation [12]. Nevertheless, as long as the time-consuming kernel matrix operations are required, it is prohibitively costly to develop a global SVM for massive spatial-temporal datasets. An efficient method uses SVMs trained locally with a small number of instances [13]. However, such an SVM is only valid locally while in many cases global and semi-global properties that are difficult to capture with local models are of primary interest.

Inspired by previous projects, we propose efficient development of semi-global SVM models valid within extended spatial regions and usable over a longer time span than those of local models. This approach is based on the observation that regional features frequently exist in spatial-temporal datasets over certain areas due to a large number of factors such as vegetation, topology, air pollution, humidity, snow, ice, and cloud coverage. Such factors are also strongly time dependent, and consequently induce semi-stationary temporal features within certain time intervals. Therefore, sub-models developed from regional datasets might remain

¹ Temple University, USA, email: zoran@ist.temple.edu

accurate for a reasonably long time or over a large area, and so can be frequently reused for reduced computational cost. The proposed method for training and reusing the semi-global SVM models will be discussed first followed by a summary of the experimental evaluations and discussion.

2. METHODOLOGY

In the simplest case where the patterns are linearly separable, an SVM looks for a separation plane which maximizes the margin. For linearly non-separable patterns, the attributes are nonlinearly mapped to a higher dimensional space such that the patterns in the new feature space are linearly separable [6]. An SVM is represented by a kernel matrix, or equivalently a set of support vectors and parameters specifying the kernel functions. A kernel matrix in a quadratic size of the number of support vectors is involved in the SVM training processes.

The support vectors of an SVM are the most important instances in the datasets from which the SVM is trained. Therefore, merging the support vectors of an SVM trained using dataset one with raw examples of another dataset results in a merged dataset which to some extent carries the properties of both datasets. The number of instances in the merged dataset would be much smaller than the total number of instances in the combined raw datasets. This observation, the starting point in this project, is aimed at the construction of semi-global training datasets of limited size for efficient learning of a series of SVMs on large spatial-temporal data series.

In spatial-temporal situations, data properties often remain semi-stationary over certain space and time intervals. Therefore, SVMs trained from neighboring local spatial datasets in a time series might frequently be similar. In such cases, merging the support vectors of a previous SVM to the adjacent raw data might be sufficient to learn a new SVM that preserves statistical properties of both regions. Based on this idea, a coarse-grained pseudo code of the proposed algorithm for learning SVMs with selective support vector propagation (SSVP) is presented below:

1. S is initialized to an empty support vector set
2. For each spatial dataset D_i , $i=1,2,\dots$, in a series
3. $D = \text{Union of } S \text{ and } D_i$
4. Train a support vector machine M_i with D
5. $S' = \text{support vector set of } M_i$
6. $S'' = \text{set of instances in } S' \text{ misclassified by } M_i$
7. $S = \text{set of instances in } S' \text{ but not in } S''$
8. End For

While the number of support vectors in each individual dataset D_i can be small, propagating all support vectors for merging with new data is likely to enlarge the merged data in time because of the inclusion of outliers that are considered support vectors. With a support vector selection method described in lines 5, 6, and 7 in the above pseudo code, the number of support vectors propagated to the next stage will be reduced significantly. More specifically, in each iteration of the training process, all the misclassified support vectors of

the resulting SVM are excluded from the support vector set to be propagated to the next training step. The effectiveness of this method will be demonstrated and discussed in the experimental results section below.

3. DATA DESCRIPTION

3.1 Synthetic Data

A series of simple synthetic datasets were generated with four cluster centers, namely $(0, 1)$, $(1, 0)$, $(0, -1)$, and $(-1, 0)$ on the 2-D plane. For each dataset, the first two clusters are labeled as positive and the other two are negative. Using different noise levels, we generated a low noise and a high noise dataset. For the low noise dataset, a randomly generated noise of normal distribution with a standard deviation of 0.1 was added to the dataset. Each dataset in the low noise series contained 100 instances with 25 instances drawn from each of four clusters. While the properties of the series, such as the location of cluster centers and the standard deviations, did not vary with time, the dataset may still be considered a spatial-temporal one in that the noise is injected in time. For the high noise dataset, the same configuration of the datasets was adopted with a standard deviation of 0.3 in noise level. A series of 500 datasets were generated for low noise and high noise cases respectively.

3.2 MISR Data

The spatial-temporal data are obtained from the MISR instrument aboard the Terra satellite of NASA's Earth observation system. MISR views Earth with nine cameras pointed at different viewing angles, each with four spectral bands. Terra orbits the Earth about 15 times per day and completes a data cycle in 16 days. For each 16-day cycle, there are 233 distinct MISR paths (orbits) [<http://eosweb.larc.nasa.gov/MISRBR>]. We analyzed four such cycles covering the entire continental U.S. from 07/17/2002 to 01/08/2003 where each cycle consisted of data over 47 paths.

The MISR Level 1B2 radiance and Level 2 aerosol optical depth (AOD) data were used for SVM model learning. We extracted 36 radiances and supplemented these with additional 82 geometric parameters resulting in 118 dimensional patterns. The learning target was the green band regional mean AOD, which was converted to binary classes of low (thin) vs. high (thick) aerosol situations as compared to the mean AOD (0.18) for the entire datasets. So, the problem of aerosol estimation from radiances was reduced to a classification problem of identifying low vs. high aerosol situations given two classes of about the same size.

The observations under cloudy conditions were removed since the existence of cloud would significantly affect the correctness of the AOD retrieval. Cases with low-quality radiance (radiometric data quality index less than 2) and with missing AOD were also removed. Therefore, the number of remaining instances used in our experiments is space and time dependent. The total number of instances in the constructed dataset of each cycle is shown in Table 3.1.

Table 3.1. Cloud-free MISR radiance and aerosol data over entire continental US (period and number of instances).

Cycle	Start Time	End Time	# of Instances
1	07/17/2002	08/01/2002	76449
2	08/01/2002	08/17/2002	65620
3	08/18/2002	09/02/2002	51869
4	12/24/2002	01/08/2003	30451

4. RESULTS AND DISCUSSION

The method of combining and reusing the SVM proposed in Section 2 was applied to both synthetic and MISR dataset series described in Section 3. A linear classifier was used for experiments on synthetic datasets and RBF classifier with unity cost of constraint violation and width parameter for experiments with MISR datasets.

4.1 Results from Synthetic Data

The first objective was to explore the rate at which the number of support vectors might accumulate in time if all vectors were propagated to the next stages. For this experiment, the proposed method for merging relevant support vector representatives with new arriving data was applied without the proposed vector selection step by setting “S” to be an empty set in line 6 of the algorithm in Section 2. The algorithm was applied to 50,000 data points partitioned into a series of 500 spatial datasets each consisting of 100 instances. The experiment was performed for low as well as for high noise synthetic data described in Section 3.1.

For a low noise situation, very few support vectors were propagated for merging with the newly arriving data independent of time. At the final 500-th time step, 10 support vectors were propagated for merging with 100 newly arrived data points and the resulting SVM was almost 100% accurate. In contrast, for high noise data, the number of propagated support vectors increased in time and soon became excessively large. In particular, unless pruned properly, about 2,200 support vectors were collected throughout the SSVP process of the initial 499 stages and they dominated over the newly arriving 100 data points. The number of the support vectors propagated at each of the 500 stages for merging with the next dataset is plotted at the top panel of Fig. 4.1. The linear trend of support vectors accumulation in this experiment was due to outliers identified as support vectors.

The selection step proposed for pruning support vectors (lines 5-7 of the algorithm described at Section 2) resolved this problem as shown at the bottom panel of Fig. 4.1. The simple selection method eliminated propagation of outliers as they were produced such that the number of propagated support vectors was much smaller (less than 140 throughout the entire process) as necessary for efficient applications in large spatial-temporal data series. While the accuracies of the final SVM obtained with and without support vector pruning on the entire dataset were similar (98.3% and 98.2%, respectively) the training and prediction with the pruning step was much faster than that without pruning.

At the bottom panel of Fig. 4.1, the slope of the curve corresponding to the number of propagated support vectors as a function of the time step is typically larger when the number of support vectors is smaller. This is expected since the position of the separation plane of an SVM is affected by the fraction of support vectors propagated from previous SVMs. The “inertia” of the separation plane, or the influence of previous datasets, increases with the fraction of propagated support vectors. On the other hand, the probability that a support vector from an earlier dataset is removed increases monotonically with time, such that the effect of a previous dataset becomes less important in the training process with the addition of new datasets. Therefore, the time interval, in which an SVM can be validly trained with the SSVP method, is determined by the competition of these two effects.

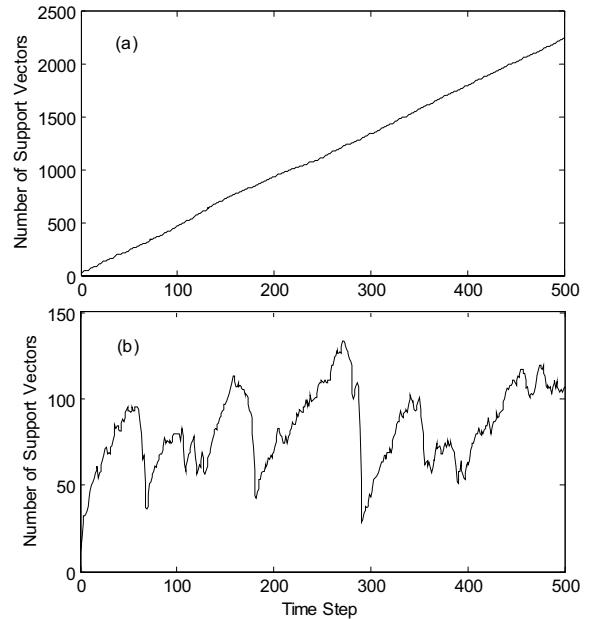


Figure 4.1. Effect of SSVP on the size of the combined datasets for high noise synthetic data. Without pruning the number of propagated support vectors in each of 500 time steps monotonically increased (top panel) while it stays small when the proposed selection step is used (bottom panel).

4.2 Results from MISR Data

The proposed method with SSVP was then applied to the MISR AOD retrieval described in Section 3. In this challenging test data properties vary both spatially and temporally with highly complex regional and temporal features. To isolate the effect of regional features of the dataset on the training of SVMs, the first set of experiments was carried out using cycle 1 aerosol data over continental U.S. (Table 3.1).

First, 47 SVMs were trained using 47 individual paths without SSVP. Each SVM was applied to predict out-of-sample AOD at all 47 paths of cycle 1. It was evident that the SVMs trained on the Eastern region (paths 4 to 27) were not applicable to Western region (paths 28 to 50) and vice versa, whereas SVMs trained on a single path on one coast were

somewhat applicable for out-of-path prediction over the same region. In particular, the average accuracy for 24 SVMs trained on a single eastern path when tested on out-of-sample data over the entire eastern region was 66%. Similarly, a single western-path-based SVM achieved 67% out-of-sample average accuracy on 23 region paths.

Next, the proposed method was applied to cycle 1 data by learning SVMs on training data corresponding to individual paths supplemented with SSVP from previously learned SVMs. In this experiment the training started from the western most path (number 50 which is California data) and proceeded east (to finish at path number 4 which is East Coast U.S. data). Again, 24 East Coast predictors were tested for out-of-sample accuracy on the entire eastern region while each of 23 West Coast predictors was tested on the entire western region. The average out-of-sample accuracy of the SVMs with SSVP on the eastern and western regions was 77% and 72% respectively, which is significantly higher than the accuracy obtained without SSVP.

The result of the first two sets of experiments provide evidence that SSVP is also beneficial for improving the prediction accuracy when data is high dimensional and the spatial relationship is complex. The next set of experiments on the MISR dataset was designed to explore the applicability of the proposed methods for modeling temporal variation in the properties of spatial-temporal datasets. For prediction on all cycles, we first applied the final predictor of the previous experiment obtained by learning local SVMs starting from the western most path (50) and moving east with SSVP. The accuracy of this predictor on all paths at all cycles is summarized at the top panel of Fig. 4.2.

As already observed, cycle 1 accuracy of this predictor was much better on the eastern region corresponding to path numbers lower than 28 (accuracy 73% vs. only 38% at west). The same predictor also had good accuracy on the eastern region when predicting at cycle 2 and 3 data (71% and 64%). The prediction accuracy on cycle 4 was consistently much worse (40%) since this data corresponds to a completely different season as compared to training period (cycle 1). The identified temporal semi-stationary property of the MISR datasets provides opportunities for developing semi-global SVM models for periods covering multiple time cycles without discarding older observations of the same region.

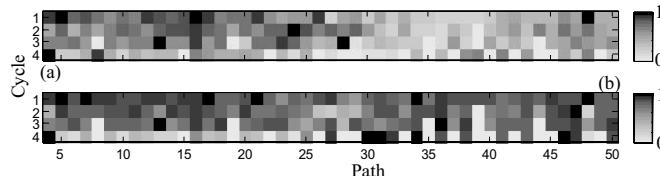


Figure 4.2. Out-of-sample classification accuracy (darker is more accurate) on Cycles 1-4 when trained on Cycle 1 data. Application of a single SVM obtained with SSVP (top panel) is compared to using 47 path-specific SVMs also constructed with SSVP (bottom panel).

Dependence of temporal semi-stationary behavior on the regional features is further examined in the experiments summarized at the bottom panel of Fig. 4.2. Here, an SVM trained on path k data of cycle 1 with SSVP from path $k+1$ at cycle 1 is used for out-of-sample classification at path k in all

four cycles. So, instead of applying a single predictor for out-of-sample prediction, 47 path-specific predictors were trained with SSVP. The resulting accuracy was much improved as can be seen by comparing top and bottom panels of Fig. 4.2. In particular, for East Cost data (paths 4-27) average accuracy in cycle 1 and 3 is improved from 73%, 71% and 64% to 85%, 73% and 75%. In addition, West Coast (paths 28-50) accuracy was 80%, 71%, 71% and 82% for cycle 1-4, respectively. Therefore, model reuse in the MISR datasets was possible both spatially and temporally.

The performed experiments suggest that the SVMs trained with SSVP should only be extended to a spatial and temporal vicinity of the training datasets and not to extremely remote spatial regions or time periods. As a result, we developed a scheme for constructing a sequence of semi-global SVMs applicable for accurate prediction to moderately large spatial and temporal regions in an efficient way (such that the number of SVMs to cover the entire dataset is minimized).

By analyzing in more details cycle 1 classification accuracy of the predictor summarized at the top panel of Fig. 4.2, it can be seen that it was highly accurate in an East Coast region covering about 10 paths followed by deteriorated accuracy as it was applied west (see Fig. 4.3). Accuracy of the global SVM trained on cycle 1 and semi-global SVMs with SSVP over shorter regions was compared next. The results over 4 cycles and propagation over 47, 24, 16, 8 and 4 paths are reported in Fig. 4.4 (propagation over 47 paths at cycle 1 corresponds to Fig. 4.3 results). The overall highest accuracy, which was even slightly better than that of the global SVM, was achieved by SSVP over blocks of 8 neighboring paths. In these experiments, SVMs trained with SSVP after every 8 paths are applied for out-of-sample prediction at the corresponding 8 adjacent paths. Consequently, only six semi-global SVMs were required to cover each data cycle.

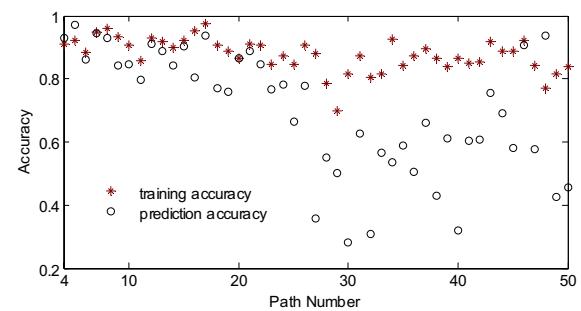


Figure 4.3. Path-specific training vs. prediction accuracy of the SVM trained with SSVP on Cycle 1. SSVP was from West to East (paths 50 to 4).

Finally, the computational efficiency of SVM training and prediction in a sequence of paths of MISR datasets with SSVP was compared to that of training a single SVM on the corresponding data. Time comparison experiments were performed on Cycle 1 data with the least number of missing values and the most balanced number of instances per path (see Table 3.1). A fixed number of 1500 instances were randomly sampled with replacement from each path (i.e. each instance may be included more than once). Both the training and the prediction CPU time was insensitive to the length of the data stream when learning in a series of local steps with

SSVP as proposed here. In contrast, for training a regular global model with merged data, the training time was a quadratic function of the number of paths N , and the prediction time increased linearly with N due to the increased size of training data (see Fig. 4.5).

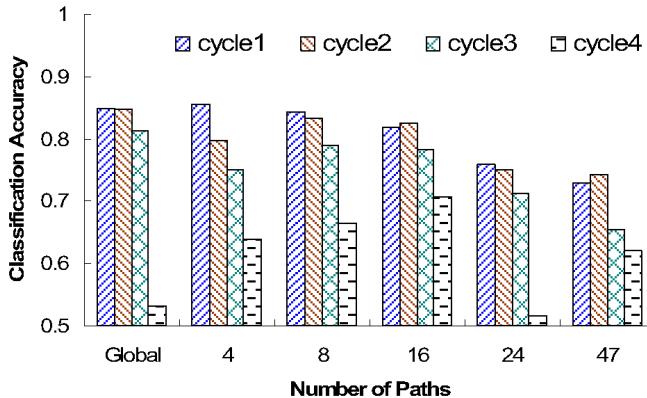


Figure 4.4. Classification accuracy on cycles 1 to 4 with global and semi-global SVMs trained on cycle 1 with SSVP over blocks of 4, 8, 16, 24 and 47 paths. Global SVM was trained with 70% instances of cycle 1.

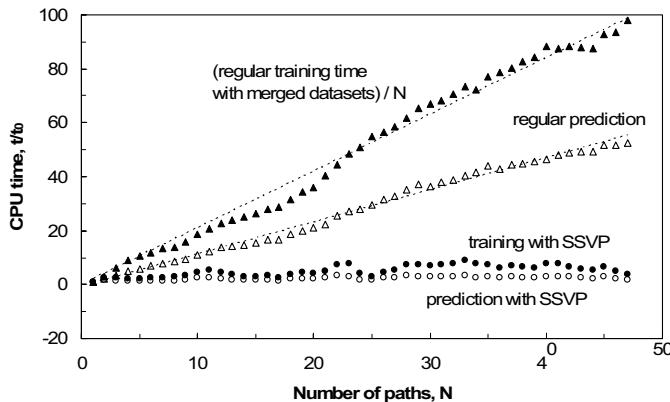


Figure 4.5. Comparison of normalized training and prediction CPU time, as a function of the number of paths N covered in training, for SVMs trained with SSVP vs. regular SVM trained using merged raw data from all paths. Training was from west to east (path 50 to 4) using the data of Cycle 1, and prediction was for each individual path only. CPU time for regular training with merged data is presented normalized by the number of paths as to show a quadratic growing curve together with a linear growing CPU time curve of regular prediction and near-constant curves obtained when training and testing with SSVP.

5. CONCLUSIONS

The proposed method of sequential SVM training with SSVP was developed for supervised learning in a non-stationary series consisting of correlated spatial datasets. The objective was to obtain accurate semi-global prediction models with low computation cost. A simple support vector selection method was applied at each propagation step to prevent monotonic growth of the number of propagated support vectors over long sequences. The validity of the proposed method was demonstrated on simple synthetic and

challenging geoinformatics spatial-temporal datasets. It was found that to predict within larger spatial and temporal neighborhoods, the SVM trained on a sequence of local spatially correlated datasets with SSVP from a neighboring region can be superior to reusing local models trained with individual datasets.

Regional features discovered using the proposed approach allowed spatial partitioning of MISR aerosol data over the continental U.S. to western and eastern regions that provided more accurate prediction results. Furthermore, regional features were also found to be semi-stationary in time, such that the regional models trained on previous datasets were reusable for accurate regional predictions at future times. The proposed model reuse method was also demonstrated as a cost-effective alternative to merging raw data when learning SVMs from large spatial-temporal data streams.

REFERENCES

- [1] Bothwell, G.W., Hansen, E.G., Vargo, R.E., and Miller K.C., ‘The Multi-angle Imaging SpectroRadiometer science data system, its products, tools, and performance’, *IEEE Trans. Geosci. Remote Sens.*, Vol. 40 No. 7, pp. 1467-1476, 2002.
- [2] Vapnik, V., *Estimation of Dependences Based on Empirical Data*: Nauka, 1979.
- [3] Peng, K., Obradovic, Z. and Vučetić, S., ‘Towards Efficient Learning of Neural Network Ensembles from Arbitrarily Large Datasets’, *Proc. 16th European Conf. Artificial Intelligence*, pp. 623-627, 2004.
- [4] Boser, B. E., Guyon, I. M. and Vapnik, V. N., ‘A training algorithm for optimal margin classifiers’, *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144-152, 1992.
- [5] Cortes, C. and Vapnik, V., *Support-Vector Networks: Machine Learning*, 20, pp. 273-297, 1995.
- [6] Schölkopf, B. and Smola, A. J., *Learning with Kernels*, MIT Press, 2002.
- [7] Vučetić, S. and Obradovic, Z., ‘Performance Controlled Data Reduction for Knowledge Discovery in Distributed Databases’, *Proc. 4th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp. 29-39, 2000.
- [8] Burges, C. J. C. and Schölkopf, B., ‘Improving the accuracy and speed of support vector learning machines’, in Mozer, M., Jordan, M. and Petsche, T., eds., *Advances in Neural Information Processing Systems 9*, pp. 375-381, MIT Press, Cambridge, MA, 1997.
- [9] Osuna, E. E. and Girosi, F., ‘Reducing the run-time complexity in support vector machines’, in Burges, C., Schölkopf B., and Smola, A. eds., *Advances in Kernel Methods: Support Vector Learning*, pp. 271–284. MIT Press, Cambridge, MA, 1999.
- [10] Osuna, E. E., Freund, R. and Girosi, F., ‘An improved training algorithm for support vector machines’, *Proc. 1997 IEEE Workshop*, pp. 276-285, 1997.
- [11] Fung, G. and Mangasarian, O. L., ‘Proximal Support Vector Machine Classifiers’, *Proc. 7th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 77-86, 2001.
- [12] Nittel, S. and Leung, K. T., ‘Parallelizing Clustering of Geoscientific Data Sets using Data Streams,’ *Proc. 16th Int'l Conf. Scientific and Statistical Data Base Management*, pp. 73-84, 2004.
- [13] DeCoste, D. and Mazzoni, D., ‘Fast Query-Optimized Kernel Machine Classification Via Incremental Approximate Nearest Support Vectors’, *Proc. 20th Int'l Conf. Machine Learning*, pp. 115-122, 2003.

Automatic term categorization by extracting knowledge from the Web

Leonardo Rigutini, Ernesto Di Iorio, Marco Ernandes, Marco Maggini¹

Abstract. This paper addresses the problem of categorizing terms or lexical entities into a predefined set of semantic domains exploiting the knowledge available on-line in the Web. The proposed system can be effectively used for the automatic expansion of thesauri, limiting the human effort to the preparation of a small training set of tagged entities. The classification of terms is performed by modeling the contexts in which terms from the same class usually appear. The Web is exploited as a significant repository of contexts that are extracted by querying one or more search engines. In particular, it is shown how the required knowledge can be obtained directly from the snippets returned by the search engines without the overhead of document downloads. Since the Web is continuously updated "World Wide", this approach allows us to face the problem of open-domain term categorization handling both the geographical and temporal variability of term semantics. The performances attained by different text classifiers are compared, showing that the accuracy results are very good independently of the specific model, thus validating the idea of using term contexts extracted from search engine snippets. Moreover, the experimental results indicate that only very few training examples are needed to reach the best performance (over 90% for the F1 measure).

1 Introduction

Term categorization is a key task in the Text Mining research area. In fact, the availability of complete and up-to-date lexical knowledge bases is becoming a central issue for automatic text processing applications, especially when dealing with rich and fast changing document collections like the Web. The maintenance of thesauri, gazetteers, and domain-specific lexicons usually requires a large amount of human effort to track changes and new additions of lexical entities. The expansion of domain-specific lexicons consists in adding a set of new and unknown terms to a predefined set of domains. In other words, a lexicon or even a more articulated structure, like an ontology [3], can be populated by associating each unknown lexical entity to one or more specific categories. Thus, the goal of term categorization is to label a lexical entity using a set of semantic themes (i.e. disciplines, domains). Domain-specific lexicons have been used in *word-sense disambiguation* [8], *query-expansion* and *cross-lingual text categorization* [10]. Several proposals to face the problem of the *automatic expansion of ontologies and thesauri* are proposed in the literature [5, 11]. In [2], the *exogenous* and the *endogenous* categorization approaches are proposed for training an automatic term classifier. The exogenous classification of lexical en-

tities is inspired by corpus-based techniques for Word Sense Disambiguation [12]. The idea is that the sense of a term can be inferred by the context in which it appears. On the other side, the endogenous classification relies only on the statistical information embedded within the sequence of characters that constitute the lexical entity itself. In [1], the authors approach this problem as the dual of text categorization. They use a set of unlabeled documents to learn associations between terms and domains and then to represent each term in the space of these documents. However, the use of a predefined collection of documents to extract knowledge can be quite limitative in highly dynamical and rich environments, like the Web, where the lexical entities involved are extremely variable, since different languages, cultures, geographical regions, domains and times of writing may coexist.

In this paper we propose an approach to term categorization that exploits the observation that the Web is a complete knowledge base that is updated continuously and is available on-line through search engines. Many recent attempts to extract the information embedded in Web documents for human-level knowledge handling are reported in the literature, such as [4] in the field of Web-based Question Answering. Nevertheless, as far as we are concerned, none of these systems is oriented to term categorization. Search engines as Google™ answer user queries returning a list of Web links along with a brief excerpt of the documents directly related to the queries. These passages, called *snippets*, represent a condensed and query-relevant version of the document contents and are designed to convey sufficient information to guide the selection of the appropriate results. Thus, snippets can provide a relevant textual context related to the lexical entities used in the query: the words in the snippets can be used to obtain a set of features for representing the query term and this representation can be used to train a classifier. The experimental results show that the information provided by the snippets is effective to attain very good classification performances, thus avoiding the overhead due to the downloads of the referred documents. Interestingly, the choice of appropriate classifier models and feature selection schemes allows us to use only few examples per class in the learning phase. Hence, the Web and, in particular, search engines can be profitably used as sources of textual corpora to automatically generate domain-specific and language-independent thesauri. This approach is general and admits any granularity in the classification. In addition, the self-updating nature of the Web can help to face the problem of thesaurus maintenance in fast evolving environments.

The paper is structured as follows. In the next section the architecture of the system is described in detail, explaining the different models used in the experiments. Section 3 reports the configuration and the results of the experiments that were aimed at assessing the system performance and at comparing the possible design choices.

¹ Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma 56, I-53100 - Siena - Italy.
{rigutini, diiorio, ernandes, maggini}@dii.unisi.it

Finally, in section 4 the conclusions and the directions for future research are presented.

2 System description

The system for term categorization is composed of two main modules as sketched in Figure 1. The *training module* is used to train the classifier from a set of labeled examples, whereas the *entity classification module* is applied to predict the appropriate category for a given input entity. Both modules exploit the Web to obtain an enriched rep-

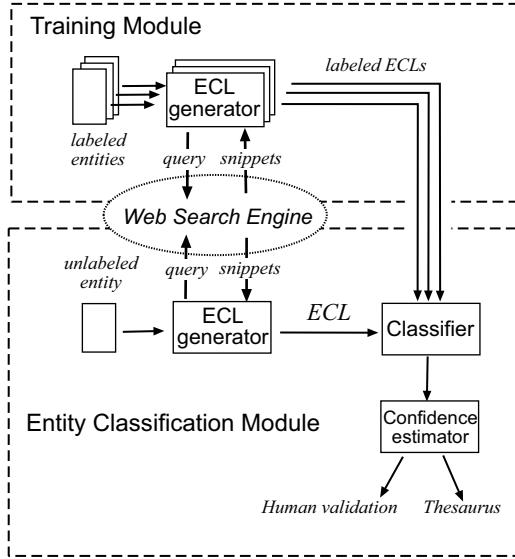


Figure 1. Block diagram of the term categorization system.

resentation of each entity. Basically the entities are transformed into queries and then the snippets obtained by one or more search engines are analyzed to build the *Entity Context Lexicon* ECL_e of each entity e :

$$e \xrightarrow{\text{Search Engine}} ECL_e.$$

In the training step, a set of labeled entities is used to train an automatic classifier to assign an ECL to one category out of a pre-defined set of classes $C = C_1, C_2, \dots, C_k$. For each labeled entity e_l , its ECL_l is computed and provided to the classifier as a training example.

To classify an unknown entity e_u , the corresponding ECL_u is obtained by querying the selected Web search engines. Then, the classifier is used to assign the resulting ECL_u to one class of the category set C . The confidence of the classifier output is evaluated to decide if the category assignment is reliable or if a human judgment is required. In the latter case, the human feedback can be used to expand the training set.

2.1 The ECL generator

This module builds the *Entity Context Lexicon* for a given entity e by analyzing the set of snippets $SN_e = \{snip_1(e), \dots, snip_s(e)\}$ obtained by issuing a query Q_e to one or more search engines. In the current implementation Q_e contains only the terms that compose the entity (e.g. Q_e is the string “New York”). Query expansion techniques will be evaluated in the future developments of the system.

The number S of snippets is a system parameter and allows us to include a different number of independent sources of entity contexts. Usually we would like to keep S as small as possible in order to include only the top-ranked results, assuming that the ranking algorithm exploited by the search engines is reliable enough to provide the most relevant and significant results in the top positions of the ranking list. By exploiting more search engines we can enforce this property since the S snippets can be obtained by exploring only the very top-ranked results. However, this advantage is achieved at the cost of a more complicated preprocessing to eliminate duplicate results. Hence, in the current implementation of the system we decided to exploit only one search engine (i.e. Google).

The ECL_e of a given entity e is simply the set of the context terms extracted from the snippets SN_e . For each word $w_k \in ECL_e$, the following statistics are stored:

- the *word count* $wc_{k,e} = \#\{w_k \in SN_e\}$;
- the *snippet count* $sc_{k,e}$ is the number of snippets in SN_e containing the word w_k , i.e. $sc_{k,e} = \#\{snip(e) \in SN_e | w_k \in snip(e)\}$.

In order to avoid the inclusion of insignificant terms, we can filter the set of the selected terms by means of a *stop-words* list. This list must be properly constructed in order to consider all the languages that are managed by the system.

2.2 The classifier module

In the system each entity e is characterized by the corresponding set of context terms, ECL_e . Hence, the term categorization task can be viewed as a text classification problem where a feature vector is associated with the ECL_e to be classified. Different term weighting schemes and classifier models can be exploited in this module. In this work, several models commonly used in text classification tasks have been tested: *Support Vector Machine (SVM)*, *Naive Bayes (NB)* and *Complement Naive Bayes (CNB)*. Moreover, we propose a prototype-based model particularly suited for this task called *Class-Context-Lexicon (CCL) classifier*, which performs the classification of each entity by evaluating the similarity between the entity and the prototype lexicons.

Support Vector Machine (SVM). The *SVM* model [6] assumes to use a mapping function Φ that transforms the input vectors into points of a new space, usually characterized by a higher number of dimensions than the original one. In this new space, the learning algorithm estimates the optimal hyperplane that separates the positive from the negative training examples for each class. The model does not require an explicit definition of the mapping function Φ , but exploits a *kernel function* $K(x_1, x_2)$ that computes the scalar product of the images of the points x_1 and x_2 , i.e. $K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$. The choice of an appropriate kernel function is the basic design issue of the *SVM* model. The training algorithm selects a subset of the training examples, the *support vectors*, which define the separation hyperplane with the maximum margin between the class and its complement.

A different *SVM* is trained for each class C_j using the ECL_e of the entities labeled with class C_j as positive examples and a subset of the other training entities as negative examples. Each ECL_e is mapped to a feature vector by adopting a specific term weighting function as described in the following. When an entity ECL_u has to be categorized, each model returns the distance between ECL_u

and its separation hyperplane. This value ranges in $[-1, 1]$ and can be considered as the similarity score between ECL_u and the class represented by the model.

Naive Bayes. The Naive Bayes classifier estimates the posterior probability of a category given the ECL . Using the Bayes rule, this value can be evaluated by estimating the likelihood of the ECL given the class:

$$P(C_j|ECL_i) = \frac{P(ECL_i|C_j)P(C_j)}{P(ECL_i)}.$$

$P(ECL_i)$ is a normalization factor constant for all categories and it can be ignored. Considering the words in the ECL as independent events, the likelihood of an unknown ECL_u can be evaluated as:

$$P(ECL_u|C_j) = \prod_{w_k} P(w_k|C_j)^{\#\{w_k \in ECL_u\}}.$$

In the training phase, each $P(w_k|C_j)$ can be estimated from the frequency of the term w_k in the class C_j .

Complement Naive Bayes. This classifier [9] estimates the posterior probability as $P(C_j|ECL_i) = 1 - P(\overline{C_j}|ECL_i)$, where \overline{C} indicates the complement of C . In this way, the probability $P(\overline{C_j}|ECL_i)$ can be easily estimated similarly as in the Naive Bayes model:

$$P(\overline{C_j}|ECL_i) = \frac{P(ECL_i|\overline{C_j})P(\overline{C_j})}{P(ECL_i)}.$$

Each $P(w_k|\overline{C_j})$ is approximated by the frequency of the term w_k in the complement class $\overline{C_j}$. This approach is particularly suited when only few labeled examples are available for each category C_j .

CCL classifier. Following the idea that similar entities appear in similar contexts, we exploited a new type of profile-based classifier. An independent classifier is trained to model each class. The profile of a given class is obtained by merging the $ECLs$ of the training entities associated with that class. Each term in the profile is associated with a weight evaluated using a given weighting function W . The profile represents the lexicon generated by all the training entities for the j^{th} class C_j and we indicate it as *Class Context Lexicon* (CCL). The CCL is simply the set of the context terms extracted from the $ECLs$ associated with the entities labeled with the corresponding class. Similarly to the case of the $ECLs$, for each word $w_k \in CCL_j$ the following statistics are stored:

- the *word count* $w_{c_{k,j}}$ counts the occurrences of w_k in the class C_j , i.e. $w_{c_{k,j}} = \#\{w_k \in CCL_j\}$. It is the sum of the $w_{c_{k,e}}$ of the $ECLs$ used to build the CCL ;
- the *snippet count* $s_{c_{k,j}}$ is the number of snippets in class C_j containing the word w_k , i.e. $s_{c_{k,j}} = \#\{snip \in CCL_j | w_k \in snip\}$. It is the sum of the $s_{c_{k,e}}$ of the $ECLs$ used to build the CCL .

When an unlabeled ECL_u is to be classified, each classifier returns a score indicating the membership degree of the ECL_u with respect to the corresponding class. First, the weights of the terms in ECL_u are evaluated using the weighting function W and then the similarity between ECL_u and each CCL_j is computed using a given similarity function.

The CCL model (as also the *SVM* classifier) exploits a function $WECL[w]$ that assigns a weight to each component w of an ECL

(or CCL). In the experiments, we tested the most commonly used weighting schemes: *binary*, *term frequency* (tf) and *term frequency-inverse document frequency* ($tfidf$). Moreover, we defined another weighting function, called *snippet frequency-inverse class frequency* ($sficf$), which combines the frequency in the snippets of each term with its distribution in each class. Being L a lexicon (ECL or CCL), $sc_{k,L}$ the snippet count of the term w_k in L and sCL the total number of snippets associated with L , the weighting function is defined as:

$$sficf(w_k, L) = \left(\frac{sc_{w,L}}{sCL} \right) \cdot \frac{1}{\#\{C | w_k \in C\}}.$$

When evaluating weights of an ECL , $sc_{w,L} = sc_{w,ECL}$ whereas in weighting terms of a CCL , $sc_{w,L} = sc_{w,CCL}$. As reported in section 3, this weighting scheme yields better performances especially when used with the CCL classifier.

Finally, the CCL classifier requires the definition of a similarity function. This function is used to compare an ECL with a CCL and yields high values when the two sets are considered similar. The most commonly used functions in automatic text processing applications are the *Euclidean similarity* and *Cosine similarity*. We also introduced a new similarity function called *Gravity*. These functions are defined as follows.

- *Euclidean similarity function.* It derives from the Euclidean distance function for vectorial spaces.

$$E(ECL_e, CCL_j) = \frac{1}{\|ECL_e - CCL_j\|}.$$

- *Cosine similarity function.* It measures the cosine of the angle formed by the two vectors.

$$C(ECL_e, CCL_j) = \frac{\langle ECL_e, CCL_j \rangle}{\|ECL_e\| \cdot \|CCL_j\|}.$$

- *Gravity similarity function.* It combines the Euclidean function, which is sensitive to the terms not shared by the two vectors, and the cosine correlation, which mainly depends on the shared terms. This function performs better when the number of terms is small (i.e. when the training set used to build the $CCLs$ contains few entities).

$$G(ECL_e, CCL_j) = \frac{\langle ECL_e, CCL_j \rangle}{\|ECL_e - CCL_j\|^2}.$$

In the previous expressions, the euclidean distance and the cosine values are computed as

$$\|ECL_e - CCL_j\| = \sqrt{\sum_w (W_{ECL_e}[w] - W_{CCL_j}[w])^2}$$

and

$$\langle ECL_e, CCL_j \rangle = \sum_w W_{ECL_e}[w] \cdot W_{CCL_j}[w],$$

where $W_{ECL_e}[w]$ and $W_{CCL_j}[w]$ are the weights of the term w in ECL_e and CCL_j , respectively.

The CCL classifier, being a prototype-based model, guarantees the modularity of the classifier when adding new categories in the domain set. In fact, each class is modeled by a CCL that is built independently of the examples in the other classes. When a new category is added, it is simply required to learn the CCL of the new class without modifying the previously computed $CCLs$. Also the Naive Bayes classifier guarantees this kind of modularity, but the experimental results showed that the CCL classifier is able to attain better performances. All the others models, instead, use negative examples in the learning phase and, therefore, they require a new training step for all the categories when a new domain is added.

3 Experimental results

We selected 8 categories (*soccer*, *music*, *location*, *computer*, *politics*, *food*, *philosophy*, *medicine*) and for each of them we searched for predefined gazetteers on the Web. Then, we randomly sampled these lists of terms in order to collect 200 entities for each class². The *soccer*, *music* and *politics* classes contain many proper names of players, coaches, teams, singers, musicians, bands, politicians, and political parties. The *location* category collects names of cities and countries, while the *computer* category mainly lists brands of computer devices and software. The *food* class contains names of dishes and typical foods, whereas in the *philosophy* category, there are the terms for various philosophical currents and concepts (e.g. casualism, illuminism and existentialism). Finally, in the *medicine* class, there are terms related to pathologies and treatments (e.g. dermatitis).

The dataset was split into two subsets, both composed by 100 randomly sampled entities: a learning collection and a test set. For each experiment we randomly partitioned the learning collection into 5 distinct training sets in order to average the results with a five-fold-validation.

The precision and recall measures were used to evaluate the system performance. Given a class C_j , the corresponding precision and recall values are defined as

$$Pr_j = \frac{TP_j}{TP_j + FP_j} \quad Re_j = \frac{TP_j}{TP_j + FN_j},$$

where TP_j is the number of the examples correctly assigned to C_j (the true positives), FP_j is the number of wrong assignments to C_j (the false positives), and FN_j is the number of examples incorrectly not assigned to C_j (the false negatives). Since the test sets for each class are balanced, we averaged these values over the set of classes (i.e. we report the *Micro Average*). Precision and recall were combined using the classical F_1 value ($F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$).

We performed a preliminary set of experiments to determine the optimal number S of snippets to be used in the construction of the *ECLs*. We found that there is no significant improvement for $S > 10$, thus we decided to perform the following tests using $S = 10$.

Using the entities in the learning collection we extracted different training sets with an increasing number of entities. We indicate as LS_M the training set containing M entities per category (i.e. a total of $8M$ entities).

The first test aimed at comparing the performances of the *CCL*-based classifier using the different weighting and similarity functions and at evaluating the influence of the size of the training set. Figure 2 reports the plots of the F_1 value for the three best configurations (we decided to report only these cases in order to improve the plot readability). The best performing model is the one that exploits the *Gravity similarity function* and the *sficf* weighting scheme. In fact, the gravity similarity shows a slightly better behavior for small learning sets. Using this configuration for the *CCL* classifier, we obtain about 90% for the F_1 value using just 5–10 examples per class. No evident improvement is achieved by adding more examples, and this result is probably due to the presence of some intrinsically ambiguous entities in the dataset. However, the system performance is very satisfactory and validates the effectiveness of the proposed approach. Table 1 collects the words with the highest score in the profile of each class using the best classifier configuration, *CCL**. It can be noticed that words in different languages (English and Italian) appear in the

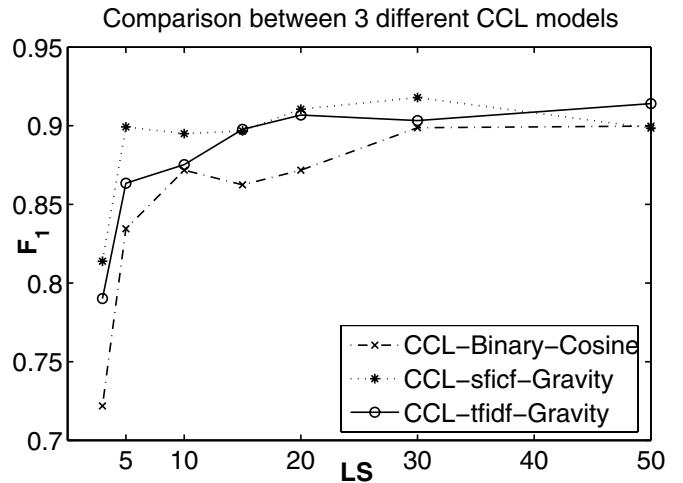


Figure 2. Plot of the F_1 values with respect to the training set size for different weighting and similarity functions in the *CCL* classifier. Each value is the average on 5 different runs exploiting different training sets.

same set, thus showing that the system can naturally adapt its behavior to a multi-lingual context.

Soccer:	league, soccer, goal, lega, campionato, rigore, attaccante, calciomercato
Politics:	partito, republican, governors, comunista, adams, hoover, clinton, coolidge
Location:	islands, india, geography, country, map, flag, tourism, city
Medicine:	atrial, infections, cavernous, hemangioma, microscopy, vascular, electron, genetic
Food:	pumpkin, bread, oil, sirloin, wheat, chicken, honey, steak
Organization:	visio, micro, silicon, drivers, laptop, virus, batteries, software
Music:	lyrics, guitar, willie, albums, chords, music, smith, fm
Philosophy:	searches, empiricus, definition, outlines, theory, knowledge, philosophical, doctrine

Table 1. The top scored words for each *CCL* in the best performing classifier.

After we individuated the best configuration for the *CCL* classifier, we compared it with the *Naive Bayes*, *Complement Naive Bayes* and *Support Vector Machine*. We used *SVM-light*³ as implementation of the SVM classifier [7] and we exploited the linear kernel that has been shown to be the best performing one in text classification tasks. In the figure 3, we notice that the *CNB* classifier attains the best performances, although they are very similar to those of the *CCL*-based classifier. Moreover, the SVM classifier shows worse performances than the other models. This model, in fact, requires a great number of support vectors to locate the best separation hyperplane. In this application, the size of training sets is quite small thus reducing the effectiveness of this model. In fact, from figure 3 we can notice that the SVM model obtains significant performance improvements when increasing the learning set size.

Globally, the performance saturates when increasing the learning

² The dataset is available at <http://airgroup.dii.unisi.it/dataset/WCD.tar.gz>

³ Available at <http://svmlight.joachims.org/>

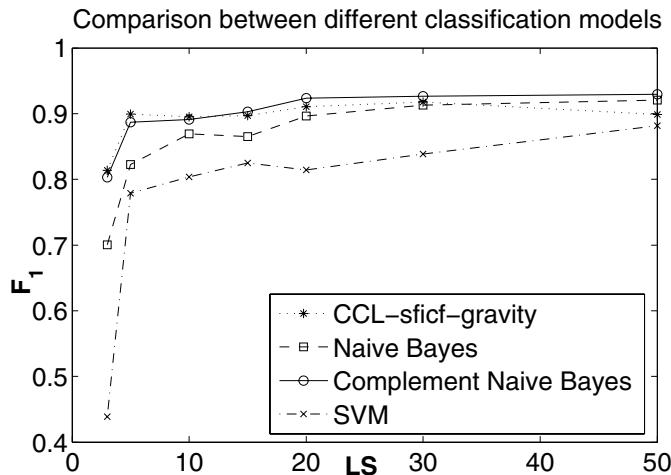


Figure 3. Plot of the F_1 values with respect to the training set size (M entities per class) for the different classifier models. Each value is the average on 5 different runs exploiting different training sets.

set size, showing that the system is able to attain good precision and recall values even with a quite limited set of examples. In particular, we individuated $M = 20$ as the best compromise between performance and learning set cardinality, there being no significant increase in the performance for larger sets.

4 Conclusions and future work

In this work we proposed a system for Web based term categorization, oriented to automatic thesaurus construction. The idea is that “terms from a same semantic category” should appear in very similar contexts, i.e. that contain approximately the same words. Based on this assumption, the system builds an *Entity Context Lexicon* (*ECL*) for each entity, which is the vocabulary composed by the words extracted from the first 10 snippets returned by Google submitting e as a query. Each *ECL* can be considered as the representation of the *term* in a feature space and an automatic classifier can be trained to categorize it into one category out of a predefined set of classes. In this work, we compared some popular classification models and we also proposed a new profile-based model that exploits the concept of class lexicon (*Context Class Lexicon*, *CCL*). The new model performs well especially when a small number of training examples is provided and it does not require a global retraining step if a new domain is added to the category set. Even if the Complement Naive Bayes model (*CNB*) yields the best performances, it does not have these two positive features. The experiments proved that with just 20 training examples per class, the system averages with over 90% of the F_1 measure. These results are very promising and we expect a further improvement after a tuning of the system design. Additional tests have been planned considering a multi-label classification of each entity and to verify the robustness of the system in “out of topic” cases, i.e. when the correct category is not present in the taxonomy.

REFERENCES

- [1] Henri Avancini, Alberto Lavelli, Bernardo Magnini, Fabrizio Sebastiani, and Roberto Zanoli, ‘Expanding domain-specific lexicons by term categorization’, in *SAC ’03: Proceedings of the 2003 ACM symposium on Applied computing*, pp. 793–797, New York, NY, USA, (2003). ACM Press.
- [2] F. Cerbah, ‘Exogenous and endogenous approaches to semantic categorization of unknown technical terms’, in *In Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pp. 145–151, Saarbrucken, Germany, (2000).
- [3] M. Ciaramita and M. Johnson, ‘Supersense tagging of unknown nouns in wordnet’, in *In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sapporo, Japan, (2003).
- [4] M. Ernandes, G. Angelini, and M. Gori, ‘WebCrow: A web-based system for crossword solving’, in *In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, (2005).
- [5] M.A. Hearst, ‘Automatic acquisition of hyponyms from large text corpora’, in *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pp. 539–545, Nantes, France, (1992).
- [6] T. Joachims, ‘Text categorization with support vector machines: Learning with many relevant features’, in *Proceedings of ECML ’98*, (1998).
- [7] T. Joachims, ‘Estimating the generalization performance of a svm efficiently’, in *Proceedings of the International Conference on Machine Learning*, (2000).
- [8] B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo, ‘The role of domain information in word sense disambiguation’, *Natural Language Engineering*, **8**(4), 359–373, (2002).
- [9] J.D. Rennie, L. Shih, J. Teevan, and D. Karger, ‘Tackling the poor assumptions of naive bayes text classifiers’, in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, (2003).
- [10] L. Rigutini, B. Liu, and M. Maggini, ‘An em based training algorithm for cross-language text categorization’, in *In Proceedings of the Web Intelligence Conference (WI)*, pp. 529–535, Compiègne, France, (2005).
- [11] N. Uramoto, ‘Positioning unknown words in a thesaurus by using information extracted from a corpus’, in *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 956–961, Copenhagen, (1996).
- [12] D. Yarowsky, ‘Word sense disambiguation using statistical models of roget’s categories trained on large corpora’, in *In Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, France, (1992).

Strategic Foresighted Learning in Competitive Multi-Agent Games

P.J. 't Hoen and S.M. Bohte and J.A. La Poutré¹

Abstract. We describe a generalized Q-learning type algorithm for reinforcement learning in *competitive* multi-agent games. We make the observation that in a competitive setting with adaptive agents an agent's actions will (likely) result in changes in the opponents policies. In addition to accounting for the estimated policies of the opponents, our algorithm also adjusts these future opponent policies by incorporating estimates of how the opponents change their policy as a reaction to ones own actions. We present results showing that agents that learn with this algorithm can successfully achieve high reward in competitive multi-agent games where myopic self-interested behavior conflicts with the long term individual interests of the players. We show that this approach successfully scales for multi-agent games of various sizes, in particular to the social dilemma type problems: from the small iterated Prisoner's Dilemma, to larger settings akin to Harding's Tragedy of the Commons. Thus, our multi-agent reinforcement algorithm is foresighted enough to correctly anticipate future rewards in the important problem class of social dilemmas, without having to resort to negotiation-like protocols or pre-coded strategies.

1 Introduction

More and more opportunities are arising where smart software agents can perform useful tasks for their owners. With these agents representing individual stakeholders, they will naturally strive to act such as to maximize the extracted utility for the stakeholder.

It is well known however that in many cases, greedy, selfish behavior by individual agents may harm the reward gained by a collective of agents as a whole, and importantly, ultimately decrease the payoff of the individual agents. This class of problems is known as 'social dilemmas', with the classic iterated Prisoner's Dilemma [15] being an example in the smallest – two player – setting, and Harding's Tragedy of the Commons [6] embodying the same problem instantiated with many players. In both cases the most valuable immediate action that a selfish agent can take lowers the value extracted by all agents collectively. The selfish agent however reaps a higher immediate reward. When all agents take this immediately most rewarding action, the resulting collective joint action has an individual payoff that is (much) worse for each individual agent than some other joint actions. Thus, in repeated play of these games, myopic selfish behavior leads for all individuals to (very) poor outcomes. The challenge is to design adaptive algorithms for individual agents that are smart and foresighted enough to allow them to quickly learn how to extract the most individual utility in repeated play of these deceptive competitive multi-agent games.

Recent work in Multi-Agent Reinforcement Learning (MARL) has proposed a number of extensions from the single agent setting to

the competitive multi-agent domain (see [3, 10, 13] for an overview). State-of-the-art MARL algorithms improve on single-agent RL approaches by incorporating models of the current opponent agents' behavior. This allows an agent to model the current "environment" including opponents, against which it has to optimize its own behavior by playing a best-response. Unfortunately, this approach encourages detrimental outcomes for social-dilemma type games.

We observe that an important aspect of competitive multi-agent games is largely ignored in MARL algorithms: the policy that maximizes payoff will typically depend on the *changing* actions of adversaries. Importantly, the adaptations of adversaries are likely to be reactions to *ones own* actions. Strategic reasoning about the expected adaptations of the adversaries can be paramount when an agent is repeatedly interacting with the same opponents, and earlier actions influence the future behavior of these adapting adversaries. As noted above, ignoring the consequences of ones actions on the development of the opponent policies can lead to detrimental outcomes, especially for settings with a social dilemma at its core.

In general, social dilemmas can be modeled by the generalized n-player iterated Prisoner's Dilemma, or nIPD [15]. In the nIPD game, every agent has a unilateral incentive to defect. However, if all agents follow this policy, then the individual reward received is lower than if all agents cooperated.

The two-player Prisoner's Dilemma (iPD) is illustrative in that in repeated play the optimal policy differs from playing the myopic best response, and it can be profitable to have a good estimation of the opponents' reactions to ones own play. In a single game, two competing agents can each choose from actions 'Cooperate' or 'Defect' (C or D), and the maximum *joint* payoff is achieved when both agents choose to Cooperate. However, an individual agent obtains a larger payoff by playing Defect, provided that the other agent plays Cooperate. From a game-theoretic perspective, playing $\{D, D\}$, the joint action with the lowest joint and individual payoff, is the dominant strategy and a Nash-Equilibrium in the single shot game.

In iterated play of the Prisoner's Dilemma game, there is no longer a clear dominant strategy, as both agents can achieve a higher aggregated and individual reward by cooperating and playing the joint action $\{C, C\}$, provided each agent has some strategic incentive to rarely unilaterally defect.

It is important to realize that this strategic incentive in Prisoner's Dilemma type games can consist of the predictable counter-moves of opponents: an intelligent adaptive adversary will react to ones own actions. The question thus becomes: which moves should one play to maximize the individual reward given the *dependent* adaptive behavior of the adversary? How will the opponents react to selfish or cooperative actions and what are good moves to play?

Strategies developed by people like Tit-For-Tat for iPD [1] enforce

¹ CWI, The Netherlands Centre for Mathematics and Computer Science, email: hoen@cwi.nl,sbohte@cwi.nl,hlp@cwi.nl

cooperative play with (smart-enough) adversaries by predictably “punishing” a defecting opponent. Defecting as a strategy is discouraged by the likely *negative future* reaction of the opponent (see also the Folk Theorem discussed in Section 2). For larger and/or less predictable games, such fixed strategies may not exist (or may be hard to determine). It may however still be beneficial for an agent to reason about the impact of its actions on the aggregated behavior of the other agents, and how this will affect the possible rewards that can be gained in future play.

We argue that current state-of-the-art MARL algorithms (reviewed in for example [13]) do not incorporate a sufficient notion of the longer term impact of their actions on the dynamics of the environment. This environment includes other adaptive agents that react to moves of their opponents as well. For example, current MARL algorithms that play the two-player iPD will universally converge to the worst possible outcome of $\{D, D\}$ when an agent plays against an opponent that uses the same MARL algorithm (self-play) [3].

The one exception we are aware of is the M-Qubed algorithm [3], which successfully learns to play $\{C, C\}$ in self play for the two-player iPD. M-Qubed adaptively switches between playing a best-response strategy and the fixed pre-coded strategy of playing one action consistently. The problem with using such pre-coded strategies is that they can be exploited [14], and in general such an approach can be expected to be unsuitable for more complex multi-agent games.

In fact, we are not aware of any state-of-the-art MARL algorithms that scale well to larger, n-player IPD games. Myopic best-response type strategies become woefully inadequate, and no current MARL algorithm has sufficient foresightedness to detect how its own behavior changes future payoffs due to other agents’ adaptive behavior.

In this paper, we make a number of important contributions: first, we refine the foresighted StrOPM MARL framework developed in [14]. This StrOPM framework builds on multi-agent Q-leaning type ideas and includes the crucial element of estimating ones opponents’ reactions into the forward reward estimations. The StrOPM algorithm in [14] successfully solves two-player two-action matrix games, including the two-player iPD. Here, we generalize the StrOPM framework such that it can be applied to games with more than two players. We develop an instantiation of this refined and generalized framework that is capable of scaling to larger size nIPD games. To deal with nIPD games, we additionally develop a generalized description of aggregate opponent policies, which we use to circumvent the state-space explosion by enabling a description of the nIPD problem in a much reduced, linear state space. We show that this approach successfully scales the StrOPM framework for the nIPD, and our StrOPM algorithm is shown to achieve a high degree of cooperation (and thus reward) even for the 9-player IPD, which was the maximum feasible game-size that ran within reasonable time. We thus argue that our generalization of the StrOPM framework is foresighted enough to correctly anticipate future rewards in the important problem class of social dilemmas, without having to resort to negotiation-like protocols or pre-coded strategies.

This result is of particular importance, as the generalized n-player iterated Prisoners Dilemma (nIPD) is a matrix game of particular interest, as noted [6]. Although the familiar 2-player iPD is a special case of the nIPD, it has been argued , that the nIPD is ‘qualitatively different’ from 2 player iPD, and that ‘... certain strategies that work well for individuals in the [2-player] iPD fail in large groups’ [2, 4, 5]. Even more than the 2-player iPD, the nIPD game is also notoriously hard for existing MARL approaches. Thus, we believe the successful generalization of the StrOPM framework to the nIPD matrix game is an important step forward for MARL algorithms.

2 Agents and Matrix games

We define the assumptions and terminology for multi-agent Reinforcement Learning in iterated play of matrix games. We refer to [2] for some well-known terms from Game Theory (GT) such as a **Nash Equilibrium** (NE), a **dominant** strategy, **pareto optimal** or **deficient** strategies, and **best-response**.

Let S denote the set of states in the game and let A_i denote the set of actions that agent/player i may select in each state $s \in S$. Let $a = (a_1, a_2, \dots, a_n)$, where $a_i \in A_i$ be a joint action for n agents, and let $A = A_1 \times \dots \times A_n$ be the set of possible joint actions.

A **strategy (or policy)** for agent i is a probability distribution $\pi_i(\cdot)$ over its actions set A_i . Let $\pi_i(S)$ denote a strategy over all states $s \in S$ and let $\pi_i(s)$ (or π_i) denote a strategy in a single state s . The generalized strategy is a **mixed strategy**, and lists for each state the probability of selecting each available action. A **joint strategy** played by n agents is denoted by $\pi = (\pi_1, \dots, \pi_n)$. Let a_{-i} and π_{-i} refer to the joint action and strategy of all agents except agent i .

We consider **matrix games**, where the payoff for each agent is defined by a set of matrices $R = \{R_1, \dots, R_n\}$. Let $R(\pi) = (R_1(\pi), \dots, R_n(\pi))$ be a vector of expected payoffs when the joint strategy π is played. Also, let $R_i(\pi_i, \pi_{-i})$ be the expected payoff for agent i when it plays strategy π_i and the other agents play π_{-i} . Let $R_i(\begin{bmatrix} a_i \\ a_{-i} \end{bmatrix})$ be the payoff for agent i playing action a_i while the other agents play action a_{-i} .

A **stage game** is a single iteration of a matrix game, and a **repeated game** is the indefinite repetition of the stage game between the same agents. While matrix games do not have state, agents can encode the previous w joint actions taken by the agents as state information, as for example illustrated in [12].

In this work, we assume that an agent can observe its own payoffs as well as the actions taken by all agents in each stage game, but only after the fact. All agents concurrently choose their actions. Adaptation of the agents’ policy, i.e. learning as a result of observed opponent behavior, only takes effect in the next stage game. Repeated games are modeled as a series of stage games with the same opponent(s). Each agent then aims to maximize its reward from iterated play of the same matrix game.

2.1 The n-player iterated Prisoner’s Dilemma

In the nIPD game, each player has a choice of two operations: either **cooperate** (C) with the other player or **defect** (D). A matrix game can be classified as a nIPD game if it has the following three properties: 1) Each player can choose between playing cooperation (C) and defection (D); 2) The D option is dominant for each player, i.e. each has a better payoff choosing D than C no matter how many of the other players choose C ; 3) The dominant D strategies intersect in a deficient equilibrium. In particular, the outcome if all players choose their non-dominant C -strategies is preferable from every player’s point of view to the one in which everyone chooses D , but no one is motivated to deviate unilaterally from D . The natural outcome of agents playing myopic Best-Response in nIPD is Defection by all agents, which is stable (a NE) but obviously Pareto-deficient.

The nIPD payoff matrix is shown in Table 1; C_i denotes the reward for cooperating with i cooperators and D_i the reward for defecting with i cooperators and $n - i - 1$ other defectors. The following conditions hold for the respective payoffs are [15]: (1) $D_i > C_i$ for $0 \leq i \leq n - 1$; (2) $D_{i+1} > D_i$ and $C_{i+1} > C_i$ for $0 \leq i < n - 1$; (3) $C_i > \frac{(D_i + C_{i-1})}{2}$ for $0 \leq i \leq n - 1$ (the payoff matrix is symmetric for each player).

		Number of cooperators among the other n-1 players				
		0	1	2	...	n - 1
player A	C	C ₀	C ₁	C ₂	...	C _{n-1}
	D	D ₀	D ₁	D ₂	...	D _{n-1}

Table 1. Structured payoffs for the (symmetrical) n-player PD

Most multi-agent learning algorithms to date have focused on an individual agent learning a (myopic) Best Response to the *current* strategies of the other agents. Play between such agents using this approach often converges, and has as a goal to converge, to a one-shot NE. However, a famous result from game theory (the **folk theorem**) suggests that the goal of reaching a one-shot NE may be inappropriate in repeated games.

The folk theorem implies that, in many games, there exists NEs for repeated games, repeated Nash-Equilibria (rNEs), that yield higher individual payoffs to all agents than do one-shot NEs, i.e. the rNE Pareto dominates the NE. Hence, in repeated games, a successful set of agents should learn to play profitable rNEs. However, since many repeated games have an infinite number of rNEs, the folk theorem does little to indicate which one the agents should play. [8] present an algorithm for computing rNEs that satisfies a set of desiderata, but how to learn these strategy online is unknown. Additionally, an agent may have preferences between rNEs and play one above the other, if allowed by its opponents.

3 The StrOPM Framework

Here, we refine the StrOPM framework of [14] and generalize it to apply to games with more than two players. In the StrOPM framework, an agent applies reinforcement learning to a state-based policy as described in Algorithm 1. At each epoch of learning, the agent adapts its policy along the gradient of increasing reward. The gradient of reward is calculated including the expected changing behavior of the opponent, as a reaction of the agent's own actions. The StrOPM algorithm tracks the changes in observed opponent policy, on a state by state basis, over time. It is assumed that these changes in the opponent behavior, at least in part, reflect reaction to actions chosen by the StrOPM algorithm. The policy of the StrOPM is then optimized with expected future reactions taken into account.

We describe the StrOPM algorithm from the perspectives of an agent i and its opponents, agents $-i$, we use this notation with subscripts to indicate policy, states, action, etc ... of the two types of agents. E.g., a_i and a_{-i} are actions of agents i and $-i$ respectively.

States. The set of states that an agent i can visit, S_i , fulfills the unichain assumption [11]: One set of “recurrent” class of states. Starting from any state in the class, the probability of visiting all the states in the class is 1. We introduce a **transition function** $T : S_i \times A_i \times A_{-i} \rightarrow S_i$ to return the next state of agent i upon playing a (joint) action from the current state.

Policies. For a state $s \in S_i$, $\pi_i(s)$ is the state-based policy of agent i , and $\pi_{-i}(s)$ is the estimate of the opponent $-i$ policies when agent i is in state s . The opponent policy is estimated online using Exponential Moving Average (EMA). The estimate of $\pi_{-i}(s)$ after observing action a_{-i} is adjusted according, for action a_i :

$$\pi_{-i,t+1}(s)(a_{-i}) = (1 - \alpha_{EMA1})\pi_{-i,t}(s)(a_{-i}) + \alpha_{EMA1}. \quad (1)$$

After this update, the policy $\pi_{-i,t+1}(s)$ is normalized to retain $\pi_{-i}(s)$ as a probability distribution.

We introduce **policy update actions**. A policy update action $pua_i(s)$ dictates whether the likelihood of an action a_i should be in-

creased for state s . Additionally, we introduce the *null* policy update action $pua_{null}(s)$ to indicate that the policy should not be changed.

Let $\pi_i(s)^{pua_i}$ be the policy achieved by applying the policy update action $pua_i(s)$ to $\pi(s)_i$. The policy $\pi(s)_i$ of agent i given $pua_i(s)$ not equal to the null action is updated according to:

$$\pi_{i,t+1}(s)^{pua_i} = (1 - \alpha_{LEARN})\pi_{i,t}(s)(a_i) + \alpha_{LEARN}, \quad (2)$$

where α_{LEARN} is the learning rate. The probabilities $\pi_i(s)(\cdot)$ for actions $a_j \neq a_i$ are then normalized to retain $\pi_i(s)$ as a probability distribution.

We wish to select policy update action $pua_i(s)^*$ in (2) that maximizes a measure of expected future payoff of the changed policy. Below, we explain how to compute this.

We introduce $\xi(\pi_{-i}(s), a_i)$ to estimate the impact of an agent i playing an action a_i on the development of the policies $\pi_{-i}(s)$ of the opponent. This function predicts the change in policy of the opponents upon playing action a_i ; i.e.

$$\pi_{-i,t+1}(s) = \xi(\pi_{-i,t}(s), a_i). \quad (3)$$

As a first implementation of this function, we take the approach that the changes in the opponent policy are, at least in part, caused by an agent's own actions. Our StrOPM implementation estimates the change in policy as a continuation of the change in policy from estimation of the opponent policy N epochs in the past, i.e. how was the opponent policy at time $t - N$? This is done for each state: for state s reached after playing action joint action $\begin{bmatrix} a_i \\ a_{-i} \end{bmatrix}$ and $T(s', a_i, a_{-i}) = s$ transitions from the current state s' to s , the change in policy for this new state is estimated as a linear extrapolation of the change in the estimated opponent policy:

$$\xi(\pi_{-i,t}(s), a_i) = \frac{\pi_{-i,t}(s) - \pi_{-i,t-N}(s)}{N} + \pi_{-i,t}(s), \quad (4)$$

where we limit ξ to $[0, 1]$. As states can encode a history of play, we judge how an opponent changes its behavior based on our choice of actions.

Thus estimating the change in opponent policies, the question becomes how to compute the value of proposed policy updates. Here, we make use of the unichain assumption that says we only need to consider loops starting from and returning to the current state to compute this value. The algorithm then updates the policy for the possible loops to increase the expected average reward.

From a state s_0 , a single loop $L(s_0)$ is defined as:

$$L(s_0) = s_0, \begin{bmatrix} a_{i,0} \\ a_{-i,0} \end{bmatrix}, s_1, \begin{bmatrix} a_{i,1} \\ a_{-i,1} \end{bmatrix}, s_2, \dots, s_{(n-1)}, \begin{bmatrix} a_{i,(n-1)} \\ a_{-i,(n-1)} \end{bmatrix}, s_0, \quad (5)$$

for a sequence of joint moves for agent i starting in s_0 , going through s_1, s_2, \dots to s_n and ending again in s_0 . Each next state reached is through a joint move; $T(s_j, \begin{bmatrix} a_{i,j} \\ a_{-i,j} \end{bmatrix}) = s_{j+1}$. All intermediate states reached in the loop are unique, and not equal to s_0 . For brevity, we denote the j -th state s_j in the sequence by $L(s_0)^j$, the j -th action pair $\begin{bmatrix} a_{i,j} \\ a_{-i,j} \end{bmatrix}$ by $L(s_0)_j$, and $L(s_0)_{j+}$ and $L(s_0)_{j-}$ the components of the j -th action pair: $a_{i,j}$ and $a_{-i,j}$. The length of the sequence $L(s_0)$, $|L(s_0)|$, is equal to n ; the number of joint actions played before the considered state is reached again.

Let $Bag(s, n) = \{L(s) | |L(s)| \leq n\}$, i.e. $Bag(s, n)$ are all the loops starting in state s with length of at most n . The probability of a particular loop $L(s)$ occurring, denoted by $Pr(L(s))$ is:

$$\begin{aligned} Pr(L(s)) = \prod_{0 \leq j < |L|} Pr(L(s)_{j+1} | \pi_{i,j}(L(s)^j)) \\ \times Pr(L(s)_{j-1} | \pi_{-i,j}(L(s)^j)), \end{aligned} \quad (6)$$

where $L(s)_j$ and $L(s)^j$ are the respective elements in the loop as defined above, $\pi_{i,j}$ and $\pi_{-i,j}$ are the respective policies at time j , evolving according to Equation (3), and $Pr(\cdot)$ denotes the probability of a specific transition along the sequence given the respective policies $\pi(s)$.

The expected reward over the possible loops can then be expressed as the weighted expected value of individual loops:

$$E(Bag(s, n)) = \sum_{L(s) \in Bag(s, n)} Pr(L(s)) \times E(L(s)), \quad (7)$$

where $E(L(s))$ is the expected average reward for each joint action of a loop starting in state s :

$$E(L(s)) = \frac{\sum_{0 \leq j < |L(s)|} V(L(s)_j)}{|L(s)|}, \quad (8)$$

where $V(L(s)_j)$ denotes the estimated value of the j -th joint action in the loop, $L(s)_j$, to agent i .

The values $V_i : A_i \times A_{-i} \rightarrow \mathbb{R}$ learns the value of a joint action to agent i ; $V_i\left(\begin{bmatrix} a_i \\ a_{-i} \end{bmatrix}\right)$ learns $R_i\left(\begin{bmatrix} a_i \\ a_{-i} \end{bmatrix}\right)$. The function V_i is updated using EMA similar to Equation 1.

Strategic Updating. Let the opponent changes in policy be estimated by $\xi(\pi_{-i}(s), a_i)$, we can then compute the policy update with highest expected payoff:

$$pua_i(s)^* = \max_i E(Bag(s, n))_{pua_i(s)}^\xi, \quad (9)$$

where $E(Bag(s, n))_{pua_i(s)}^\xi$ denotes the expected average reward for the possible loops of at most length n , starting in state s , after effecting policy update action $pua_i(s)$ and taking $\xi(\pi_{-i}(s), a_i)$ as the estimated opponent adaptation. Our StrOPM algorithm thus obtained is outlined in Algorithm 1.

4 Experiments

We demonstrate StrOPM for nIPD games of different sizes: from the standard two-player IPD to nIPD games with up to nine players.

For the two-player IPD, the payoff matrix is shown in Figure 1a, inset. We let each agent encode as states the last joint action played; $\{C, C\}$, $\{C, D\}$, $\{D, C\}$, $\{D, D\}$.

This type of state representation, along with the detailed modeling of each individual opponent policy, cannot be scaled indiscriminately for an increasing number of agents (for n players the number of states per joint action scales as 2^{n-1}). We observe however that for nIPD with more than two players, we can reduce the state space representation: even though the opponent payoff contributions C_{-i} and D_{-i} in Table 1 can be different for individual opponents $j \in -i$, their contribution to the reward gradient is summed, and each agent playing this game only experiences this summed gradient.

Thus, we can choose a simpler state representation where we model the behavior of the aggregated opponents. We choose policy update actions using the estimates of the aggregated opponent behavior, and determine the reward gradient through the expected global change in behaviors of the opponents as follows: For each agent, we encode as states the number of other agents that have cooperated in

Algorithm 1 StrOPM

```

1: Initialize  $\pi_i(s)$ ,  $\pi_{-i}(s)$  for all  $s \in S_i$  and  $V$  for all joint actions. Set the initial state.
2: Do in each epoch sequentially for each agent  $i$  in state  $s$ :
3: loop
4:   calculate the highest valued  $pua_i(s)^*$  using  $E(Bag(s, n))_{pua_i(s)}^\xi$  as value for the individual policy update actions.
5:   Update policy  $\pi_i(s)$  using the highest valued policy update  $pua_i(s)^*$  action.
6:   Play action  $a_i \in A_i$  based on the chosen policy update action  $pua_i(s)^*$ . StrOPM plays action  $a_i$  for chosen policy update action  $pua_i(s) = pua(s)^*$  if  $pua^*$  is not the null policy update action. Otherwise choose the action according to  $\pi_i(s)$ .
7:   Receive reward  $R_i([a_i a_{-i}])$  for the joint action determined by agents  $-i$ .
8:   Update the estimate of the reward for the joint action  $V([a_i a_{-i}])$  and the opponent policy  $\pi_{-i}$ .
9:   set the current state to  $T(s, \begin{bmatrix} a_i \\ a_{-i} \end{bmatrix})$ .
10: end loop

```

the last epoch; i.e. for n players there are $n - 1$ states that capture whether 0, 1, to $n - 1$ other agents cooperated in the last epoch. For each state s , separately for action C and D , each agent maintains an estimate of aggregate opponent policy $\pi_{-i}(s)$. The function ξ in Equation (4) then effectively captures the likelihood of either i) staying in the same state, ii) going to a state with more cooperators, or iii) moving to a state with less cooperators as a consequence of playing either action.

The function V learns the payoff Table 1. As for the two agent case, $\pi_i(s)$ encodes for each state the probability of agent i playing C in that state. A learning rate of 0.01 was used for all the EMA equations of Section 3. The StrOPM algorithm looks back $N = 10$ epochs in Equation 4. Additionally, an agent using StrOPM had a $\epsilon = 0.01$ probability of taking an exploration move in each epoch to ensure all states are sufficiently sampled in play.

StrOPM in the 2-player IPD. We present results of the StrOPM algorithm for self-play in the two-player IPD (Figure 1a, solid line), and for more myopic agents in self-play (Figure 1a, dashed line). The myopic agents used a restricted version of StrOPM, MOPM, that was constructed by using StrOPM and setting $\xi(\pi_{-i}, a_i) = \pi_{-i}$ (i.e., the agent assumes that its adversaries do not adapt in response to its own actions). MOPM thus mimics the best-response type strategies most state-of-the-art MARL algorithms employ.

As can be seen in Figure 1a, the StrOPM learner in self-play converges to playing the (optimal) Cooperate-Cooperate ($\{C, C\}$) strategy with reward 0.35 (solid line). The full cooperation equilibrium is reached as the StrOPM learner estimates that unilaterally defecting leads to states where more and more defection is expected. Additionally, the full cooperation state is expected to lead to more cooperation. In contrast, agents using MOPM in self-play quickly “learn” that cooperation is risky and move to full defection (dashed line).

A closer study of the state-based policy of the StrOPM players reveals that the agents in self-play exhibit a learned Tit-For-Tat strat-

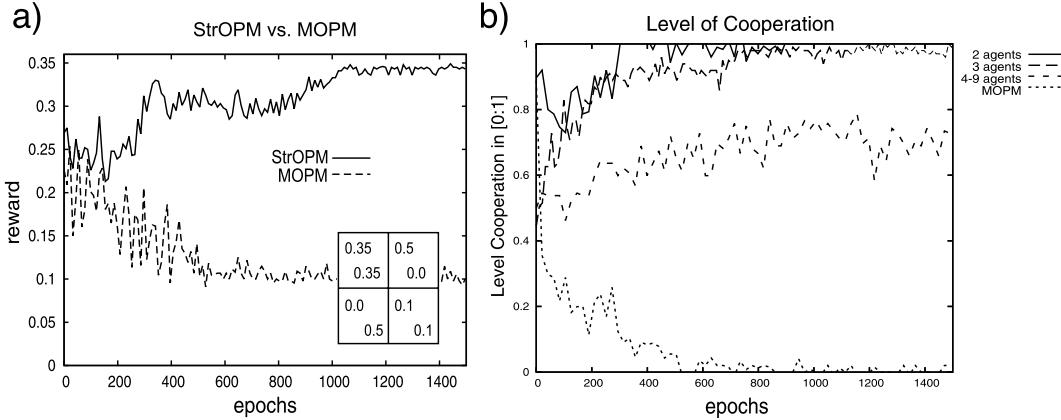


Figure 1. a) Inset: payoff matrix for a two player iPD game. Graph: average payoff for two StrOPM agents and two MOPM agents playing the iPD. b) Cooperation, as % of all agents cooperating, for 3 to 9 players in nIPD using StrOPM.

egy [1]. The Tit-For-Tat algorithm plays C until the opponent defects upon which a D is played followed by again C until the next defection to encourage cooperation. With states encoded as outlined above, the policy is to play C in state $\{C, C\}$, play D from state $\{C, D\}$, and C from $\{D, D\}$. The StrOPM agents learn this strategy and play C to cooperate and reach a good equilibrium, except for occasional exploratory moves. At the same time the StrOPM players also evolve a “threat” state where defection is retaliated in order to guard against exploitation by the opponent. Note that StrOPM also learns to play optimally in less difficult 2-player matrix games, like chicken, rock-paper-scissors, and matching pennies (not shown).

StrOPM playing nIPD. The real challenge for playing strategic games like nIPD, is to successfully learn to cooperate with many players. In Figure 1b, we show results for three and more agents playing the nIPD. The x-axis shows the number of epochs learned, while the y-axis shows the average number of cooperators. For three agents, the StrOPM algorithm learns full cooperation. For four up to nine agents, the nIPD is no longer solved in full. The agents as a collective learn to cooperate approximately 70% of the time. Nine-player IPD was the maximum game-size for which we found it feasible to still compute the loops as defined in Equation 5 for the Bags used in the algorithm.

In Figure 1b, bottom line, we show nIPD results for agents using myopic MOPM described above for three agents, to model agents that do not reflect on the impact of their actions; this short-sighted exploitation of the gradient leads to full defection of the agents. This illustrates that as for the two player iPD, classic MARL algorithms that ignore the impact of ones own actions on other agents will, for each state, increase the probability of playing defect, leading to the well known Tragedy of the Commons: universal defection of all agents.

5 Conclusions

We make a number of contributions in this paper: first, we refine the StrOPM MARL framework developed in [14], and generalize it so that it can be applied to matrix games with more than two players. We show that our implementation of this generalized framework successfully scales to larger size nIPD games. To this end, we develop a generalized description of aggregate opponent policies, which allows us to circumvent the state-space explosion and enables us to describe the nIPD problem in a much reduced, linear state space. We show that using this state-space description, our generalized StrOPM algorithm allows an agent to learn profitable strategies for long term behavior in generalized n-player IPD for games with up to nine players. The

successful scaling of the framework is of particular interest because, as noted earlier, the n-player iterated Prisoners Dilemma is a game that can represent many important and hard real life problems.

One important novel component of this work is the concept of learning the *aggregate* changes in opponents’ policies due to ones own actions. This is an idea that can in fact be incorporated into the quickly growing literature on MARL. One can foresee more and more complex nested opponent models [7] to extract every bit of reward from complex games like nIPD. Although perfectly learning about an opponent while at the same time perfectly learning to adjust oneself is problematic [9], there is still much scope to be “smarter” than your opponents.

REFERENCES

- [1] R. Axelrod, *The evolution of cooperation*, Basic Books, New York, NY, 1984.
- [2] A.M. Colman, *Game Theory and Experimental Games, The Study of Strategic Interaction*, volume 4 of *International Series in Experimental Psychology*, Pergamon Press, Oxford, 1982.
- [3] J. W. Crandall and M. A. Goodrich, ‘Learning to compete, compromise, and cooperate in repeated general-sum games’, in *Proc. 22nd ICML*, (2005).
- [4] N. S. Glance and B. A. Huberman, ‘The outbreak of cooperation’, *Journal of Mathematical Sociology*, **17**(4), 281–302, (1993).
- [5] N. S. Glance and B. A. Huberman, ‘Dynamics of social dilemmas’, *Scientific American*, 76–81, (1994).
- [6] Garrett Hardin, ‘The tragedy of the commons’, *Science*, **162**, 1243–1248, (1968).
- [7] J. Hu and M.P. Wellman, ‘Online learning about other agents in a dynamic multiagent system’, in *Proc ACM Conf. on Autonomous Agents*, pp. 239–246, (1998).
- [8] Michael L. Littman and Peter Stone, ‘A polynomial-time Nash equilibrium algorithm for repeated games’, in *Proc. 4th ACM Conf. on Electronic Commerce*, pp. 48–54, (2003).
- [9] J. H. Nachbar and W. R. Zame, ‘Non-computable strategies and discounted repeated games’, *Economic Theory*, **8**, 103–122, (1996).
- [10] R. Powers and Y. Shoham, ‘New criteria and a new algorithm for learning in multi-agent systems’, in *NIPS*, (2004).
- [11] Martin L. Puterman, *Markov Decision Process*, John Wiley and Sons, Inc., New York, 1994.
- [12] T. Sandholm and R. Crites, ‘Multiagent reinforcement learning in the iterated prisoner’s dilemma’, *Biosystems*, **37**, 147–166, (1995).
- [13] Yoav Shoham, Robert Powers, and Trond Grenager, ‘Multi-agent reinforcement learning: a critical survey’, in *AAAI Fall Symposium on Artificial Multi-Agent Learning*, (2004).
- [14] P.J. ’t Hoen, S.M. Bohte, and J.A. La Poutré, ‘Learning from induced changes in opponent (re)actions in multi-agent games’, in *AAMAS’06*, (2006). to appear, available as technical rapport SEN-E0513.
- [15] Xin Yao and Paul J. Darwen, ‘An experimental study of n-person iterated prisoner’s dilemma games’, in *Evo Workshops*, pp. 90–108, (1994).

6. Natural Language Processing

This page intentionally left blank

MSDA: Wordsense Discrimination Using Context Vectors and Attributes

Abdulrahman Almuhareb¹ and Massimo Poesio^{1,2}

Abstract. We present MSDA (Major Senses Discovery Algorithm) – a development over the context vector approach to (noun) sense discrimination [20, 24] that uses attributes and values instead of word features to cluster contexts, and does not require for the number of senses to be fixed beforehand. The algorithm achieves a precision of 89% on a dataset including both ambiguous and non-ambiguous nouns, twice that of previous algorithms.

1 INTRODUCTION

Arguably, the most difficult part of the task of acquiring lexical and ontological knowledge from text [2, 6–8, 13, 16, 18, 23, 24] is the identification of a word’s senses—i.e. trying to discover that a word such as *palm* can be used to express both the concept “the inner surface of the hand” (WordNet 2 sense 1), henceforth **palm1**, and the concept “any plant of the family Palmae” (WordNet 2 sense 3), henceforth **palm3** [5, 16, 20, 22, 24]. Not the least among the difficulties is the fact that whereas **major sense** distinctions like the one just mentioned can generally be made reliably, other distinctions—e.g. that between **palm1** above and the derived sense of “a linear unit based on the length or width of a human hand” (WordNet 2 sense 2) are more difficult [12].

Our own work on the acquisition of lexical knowledge has been motivated by the consideration that knowing about concept **attributes**³—e.g. knowing that **ships** have **captains**—is an essential aspect of the rep-

resentation of concepts in AI [4, 26], Linguistics [21] and Psychology [15]. We developed methods for extracting from the Web candidate concept attributes, and semantic classifier to select those that fit into a linguistically and philosophically motivated classification of attributes, finding that this leads to improvements at concept clustering.

The motivation for the work discussed in the present paper is the intuition that such information about attributes ought to help with sense discrimination as well, in that different senses of a word ought to be associated with different attributes. Thus, for example, instances of **palm1** are more likely to be connected with a **wrist** than instances of **palm3**; conversely, the latter will be more likely to have **coconuts**. (Of course, there will also be attributes in common. E.g. instances of both **palm1** and **palm3**, being physical objects, will have a length—although we conjectured that these would not be the most distinctive attributes.) In addition, we explicitly focus on the task of identifying major senses instead of all sense distinctions as identified, say, in WordNet.

The structure of the paper is as follows. After a brief overview of relevant literature on sense discrimination and on using attributes for concept clustering, we introduce our new algorithm for sense discrimination, MSDA, and present the experiment we used to evaluate it. Results and discussion follow.

2 BACKGROUND

2.1 Wordsense discrimination with context vectors

The old observation that context is required to determine a word’s meaning led, e.g. Miller and Charles [14] to suggest that similar meanings are often used in similar contexts. Schütze [24] implemented this intuition by extending his earlier vector space model [23] to identify senses of words by using two different types

¹ University of Essex, UK, email: aalmuh & poesio at essex.ac.uk.

² University of Trento, Italy.

³ The term **attribute** is used informally here to indicate the type of relational information about concepts that is expressed using so-called **roles** in Description Logics [4]—i.e. excluding **IS-A** style information (that cars are vehicles, for instance). It is meant to be a more restrictive term than the term **feature**, often used to indicate any property of concepts, particularly in Psychology. We are carrying out a systematic analysis of the sets of features used in work such as [25] (see Discussion).

of vectors: **first-order vectors** representing contexts in terms of neighboring words of the target word, as in his original model; and **second-order vectors**, used to represent a context as the centroid of the first-order vectors for the words occurring in that context—which amounts to using word vectors as features. The context vectors for a word are then clustered using agglomerative clustering in a predefined number of sense clusters. Schütze reported an accuracy of between 72% and 90%, but pointed out that his algorithm should be tested using more natural ambiguous words. Pedersen and Bruce [17] developed a second method for identifying senses, in which contexts were directly represented with first-order vectors, i.e. using words themselves as features. These first-order vectors were then clustered using agglomerative cluster and the EM algorithm.

Purandare and Pedersen [20] (henceforth: P&P) compared these two methods, also looking at different combinations of clustering and features, and concluded that first-order vectors and agglomerative clustering are the best choice when the sample data is large (e.g. about 4,000 contexts per word), while second-order vectors and the Repeated Bisects algorithm are the best when the sample data is small (e.g. around 50-200 contexts per word). P&P [20] also developed a (publicly available) software package called **SenseClusters** implementing a variety of context representation models include first-order and second-order vectors with and without dimensionality reductions. The software also provides several clustering algorithms including agglomerative, partitional, and Repeated Bisects [11] algorithms.

2.2 Mining concept attributes from the Web

As said earlier, most theories of concepts are based on the assumption that concepts are characterized in terms of semantic attributes or, more generally, features, but no previous work on the acquisition of lexical or ontological knowledge by concept clustering attempted to identify such semantic properties. In our own earlier work [2, 18] we developed methods for extracting candidate attributes from the Web using patterns as done, e.g. by Hearst [10] to identify **is-a** relations or Poesio et al. [19] to identify **part-of** relations. For example, Hearst [10] acquired information about hyponymy (**is-a** links) by searching for instances of patterns such as

NP {, NP}* or other NP

(as in, e.g. *bruises broken bones and other INJURIES*). In our own work, we used the pattern

"the * of the C [is|was]"

(suggested by e.g. Woods [26] as a test for ‘attribute-hood’) to search for candidate attributes of concept *C* in the Web (finding e.g. instances such as *the color of the car is*) and a separate pattern for identifying **values** of such attributes (such as *red*). We showed in [3] that using only such attributes and values results in better concept descriptions for the purposes of clustering than collecting all relations in which a concept occurs. We then trained a classifier using morphological information and information from the Web for filtering such candidate attributes by assigning them to one of five classes according to a scheme derived from the work of Pustejovsky [21] and Guarino [9]. We showed in [18] that a binary classifier filtering non-attributes this way (i.e. eliminating attributes not recognized as belonging to one of the five classes) leads to significant improvements in concept clustering.

The conjecture motivating the present work was that such methods for identifying attributes and values could work better as a feature selection mechanism than the methods used by Schütze or P&P for the purposes of identifying distinct context clusters, each representing a separate sense.

3 A NEW SENSE DISCRIMINATION ALGORITHM

Our new sense discrimination algorithm, MSDA (Major Senses Discovery Algorithm) improves over the algorithms implemented in SenseClusters in three respects: (i) it uses attributes and values as semantic features used to identify different senses; (ii) it does not require to predefine the number of senses; and (iii) it returns a flag specifying whether context information provides enough evidence to decide on a number of senses.

The pseudo-code for MSDA is shown in Figure 1. The algorithm begins by computing the set *C* of context vectors *c*. The algorithm is designed to work with large amounts of data such as those that can be extracted from the Web (we use up to 10,000 contexts per word, see below); hence, on the basis of the Purandare and Pedersen results, it uses first-order vectors, one for each occurrence of target noun *w*. After collecting the contexts, an iteration begins. Each loop consists of two phases. In **Phase 1**, the contexts in *C* are clustered into *n* clusters (at the beginning *n* = 2; it is incremented by 1 at each iteration). Against the recommendations of Purandare and Pedersen, we used Repeated Bisects for clustering, with a criterion function that tries to maximize the internal similarity of each cluster [11]. We did not use agglomerative clustering because we found that it produces clusters

with varying sizes: some clusters contain very large number of instances, while other ones a very small number, which is not adequate in our case because we are only looking for major senses, that should be frequent in the data.

Input: A set of contexts C ; the thresholds Φ_1 and Φ_2 ; and m , the number of features to be compared.

$n = 2$; $f \leftarrow \text{true}$;

repeat

Phase 1:

 partition C in n clusters $k_1 \dots k_n$ by repeated bisections;

Phase 2:

foreach $i=1..n$ find the m most frequent attributes and values in cluster k_i ;

$k_i, k_j \leftarrow$ the two clusters such that $\text{sim}(k_i, k_j)$ is max;

if $\text{sim}(k_i, k_j) \geq \Phi_2$ **then**

$n \leftarrow n-1$; **halt**;

if $\Phi_1 < \text{sim}(k_i, k_j) < \Phi_2$ **then**

$n \leftarrow n-1$; $f \leftarrow \text{false}$; **halt**;

$n \leftarrow n+1$;

end repeat

Output: n , the number of the discovered senses; f , a flag indicating if MSDA is certain or not; and the top frequent attributes & values related to each sense.

Figure 1: Pseudo-code for MSDA.

In **Phase 2**, attributes and values are used to discriminate between clusters. They are extracted from the contexts in each cluster k_i using the methods developed in our previous work, and the top m most frequent attributes / values for each cluster are used to decide which clusters k_i and k_j are most similar (using the cosine similarity function on binary values). At this point, a decision is made whether to halt or to try to partition C into more clusters. The decision is made by comparing $\text{sim}(k_i, k_j)$ with two thresholds. If the similarity between the top m features of the two most similar clusters is \geq threshold Φ_2 , MSDA halts, and concludes that there are ($n - 1$) major sense(s). If the similarity is less than or equals a second threshold Φ_1 ($\Phi_1 < \Phi_2$), n is incremented and the next iteration is attempted. If the similarity falls in between these two thresholds, the algorithm reports that it is uncertain about the results, and there are at least ($n - 1$) major sense(s) for the target noun.

The intuition behind the decision procedure is as follows. The higher threshold Φ_2 is used to decide when two sets of attributes / values are similar enough that two senses should be considered the same. Suppose that noun w only has one sense. Then the two initial clusters should share many attributes / values, hence the similarity between them should be $\geq \Phi_2$. The lower threshold Φ_1 is used to decide when two clusters

share so few features that they belong to two different senses. Suppose noun w has two or more clearly distinct senses. Then the sets of most distinctive attributes should be very different, and the similarity should be $\leq \Phi_1$. The algorithm will try to find more distinctive senses (but it will return $n-1$ if it ends up splitting a cohesive cluster). In other words, MSDA continues partitioning the contexts until it finds two very similar clusters. Finally, if the similarity falls in between the two thresholds, the evidence is not strong enough to give a certain conclusion. The algorithm also halts reporting an uncertain conclusion if there are less than m distinct features for any of the clusters in the set.

MSDA compares valid attributes (e.g. qualities, activities, and parts) and valid values (e.g. *fast*, and *slow* for the quality *speed*) only; invalid attributes are filtered out using the 2-way attribute classifier proposed in [18], and invalid values are filtered out using methods described in [1]. Note also that to eliminate the effect of these semantic features on the clustering, these features are hidden in Phase 1, and only used in the comparison in Phase 2.

4 EVALUATION

We now discuss how we evaluated MSDA, and compared it with Purandare and Pedersen's SC.

4.1 Test data

In order to facilitate the comparison with previous work, we tested MSDA using a combination of unambiguous, naturally ambiguous, and pseudo-ambiguous words as done by Schütze and others. Our natural words included two ambiguous nouns used by Schütze (**capital** and **interest**), the one used by Rapp [22] (**palm**), and three nouns that are unambiguous (according to WordNet) and frequent: *brigade*, *hydrogen*, and *ship*. These three unambiguous nouns were also used to create three artificially ambiguous words: *brigade_hydrogen_ship*, *hydrogen_ship*, and *brigade_hydrogen*. ‘Contexts’ for these words were created by replacing e.g. every occurrence of *hydrogen* and of *ship* with *hydrogen_ship*, as done by Schütze.

4.2 Data collection

We collected from the Web about 10,000 contexts of occurrence for each natural noun in the dataset. For the pseudo-ambiguous nouns, we included a balanced number of contexts for each of the natural nouns that make them: e.g. the contexts for *hydrogen_ship* contain 5,000 contexts related to *hydrogen* and 5,000 contexts related *ship*. The context vectors were built using

a window of size 50 around the target noun. In phase 2, attributes and values are extracted from each context using the patterns discussed in our previous work.

We experimented with different values for Φ_1 , Φ_2 , and m (the number of most frequent attributes / values for each cluster). The best results were obtained with $\Phi_1 = 0.15$, $\Phi_2 = 0.30$, and $m = 20$.

4.3 Comparisons

We also used the data collected as above to test SenseClusters on the same dataset using Purandare and Pedersen's [20] recommendations and choices for clustering a large data sample. Again, about 10,000 contexts were used for each noun. The context vectors were constructed using a scope of size 5 and a window of size 40 (i.e. 20 in either side of the target noun). As done by P&P, we fixed the number of clusters to 7, but discarded clusters with few instances, less than 2% of the total number of instances, the threshold used by P&P. (We also tried a higher threshold (20%) and obtained similar results.) First-order contexts were used, and clustered using the agglomerative algorithm. Less frequent neighboring words with overall frequency of less than 2 were ignored. Frequencies were weighted using the log-likelihood function using the threshold 3.841.

4.4 Results

The results using SenseClusters (SC) and MSDA are shown in Table 1. The second column lists the number of major senses found in the literature for the natural ambiguous nouns (and based on WordNet for the unambiguous nouns); the third column shows the number of senses found using SC; the fourth the number of senses found using MSDA.

Noun	Major Senses	SC	MSDA
brigade	1	1 ✓	1 ✓
hydrogen	1	3 ✗	1 ✓
ship	1	1 ✓	1 ✓
brigade_hydrogen	2	2 ✓	2 ✓
hydrogen_ship	2	3 ✗	2 ✓
brigade_hydrogen_ship	3	2 ✗	3 ✓
capital	2	1 ✗	2 ✓
interest	2	1 ✗	1 ✗
palm	2	2 ✓	2 ✓

Table 1: Results with SC and MSDA

As can be seen from the Table, MSDA works very well: it identifies the correct number of senses for 8 out of 9 nouns, 89%. By contrast, SenseClusters only

found the correct number of major senses for 4 out of 9 nouns, 44%.

4.5 Semantic features related to each sense

Table 2 shows the 4 most frequent semantic features (attributes and values) for each sense of the five ambiguous nouns found using MSDA. The features are ordered by frequency. Attribute features are preceded by '[a]', while value features are preceded by '[v]'. The first column in the table shows a manually assigned label for each sense.

Label	Top 4 Frequent Features
brigade_hydrogen	
brigade	[v]fire, [v]Texas, [v]1st, [a]commander
hydrogen	[v]liquid, [v]amide, [v]element, [v]fire
hydrogen_ship	
hydrogen	[v]liquid, [v]amide, [v]element, [v]molecular
ship	[a]captain, [a]name, [a]safety, [a]owner
brigade_hydrogen_ship	
brigade	[v]Texas, [v]1st, [v]3rd, [a]commander
hydrogen	[v]liquid, [v]amide, [v]element, [v]molecular
ship	[v]fire, [a]captain, [a]name, [a]side
capital	
assets	[v]fixed, [v]variable, [a]value, [v]circulating
district	[a]streets, [a]transfer, [a]removal, [a]heart
palm	
hand	[a]size, [v]right, [v]left, [a]base
tree	[v]date, [v]coconut, [v]oil, [v}sago
interest	
Common features: [a]extent, [a]holder, [a]transfer, [a]value, [v]controlling, [v]partnership, [v]security	

Table 2: Top 4 semantic features for each sense found by the MSDA algorithm.

Most of the attributes and values in the table are clearly related to the named sense. For example, the *assets* sense of *capital* includes the attribute *value* and the values *fixed*, *variable*, and *circulating*, which are clearly related to *assets*; while most of the attributes *streets*, *transfer*, *removal*, and *heart* are more related to the *district* sense of *capital* than to the *assets* sense.

The table also shows the data for the noun *interest* where MSDA failed to find the correct number of major senses: In the first iteration of Phase 2, MSDA found 7 common features (shown in the table) among the top 20 features of the two clusters of the *interest* contexts which gives a similarity score of 0.35 that is above the threshold Φ_2 .

5 DISCUSSION AND CONCLUSIONS

This work suggests using semantic features such as

attributes and values is not only a good way to build concept descriptions for the purposes of clustering, but also to discriminate between noun senses. This work however raises a number of questions that we hope to explore in further research, such as whether it is possible to come up with a clear enough repertoire of ‘major senses’ for some words that we will be able to assess whether in fact the algorithm is always successful at the task. (Unfortunately, the distinction between polysemy and homonymy is notoriously difficult to draw – see again Kilgarriff’s paper.)

ACKNOWLEDGMENTS

Abdulrahman Almuhareb is supported by King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia.

REFERENCES

- [1] Almuhareb, A. (2006) *Attributes In Lexical Acquisition*, PhD Thesis, Department of Computer Science, University of Essex.
- [2] Almuhareb, A. & Poesio, M. (2004) Attribute-based and value-based clustering: An evaluation. In *Proceedings of EMNLP*. Barcelona.
- [3] Almuhareb, A. & Poesio, M. (2005) Finding Attributes in the Web Using a Parser. In *Proceedings of Corpus Linguistics*. Birmingham.
- [4] Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. (Eds.) (2003) *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press.
- [5] Caraballo, S. A. (1999) Automatic construction of a hyponym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*. College Park, MD, USA, 120-126.
- [6] Curran, J. R. & Moens, M. (2002) Improvements in Automatic Thesaurus Extraction. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition*. Philadelphia, PA, USA, 59 - 67.
- [7] Dorow, B. & Widdows, D. (2003) Discovering Corpus-Specific Word Senses. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest, Hungary, 79-82.
- [8] Grefenstette, G. (1994) *Explorations in Automatic Thesaurus Discovery*, Boston, USA, Kluwer Academic Publishers.
- [9] Guarino, N. (1992) Concepts, attributes and arbitrary relations: some linguistic and ontological criteria for structuring knowledge bases. *Data and Knowledge Engineering*, 8, 249-261.
- [10] Hearst, M. A. (1998) Automated Discovery of Word-Net Relations. IN FELLBAUM, C. (Ed.) *WordNet: An Electronic Lexical Database*. MIT Press.
- [11] Karypis, G. (2003) *CLUTO: A clustering toolkit*. Technical Report #02-017 [online]. Department of Computer Science, University of Minnesota, Minneapolis, MN. Available from: <http://www-users.cs.umn.edu/~karypis/cluto/> [Accessed 27/09/2005].
- [12] Kilgarriff, A. (1997) I don't believe in word senses. *Computers and the Humanities*, 31(2), 91-113.
- [13] Lin, D. (1998) Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*. Montreal, Canada, 768-774.
- [14] Miller, G. A. & Charles, W. G. (1991) Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1), 1-28.
- [15] Murphy, G. L. (2002) *The Big Book of Concepts*, The MIT Press.
- [16] Pantel, P. & Lin, D. (2002) Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. Edmonton, Alberta, Canada, 613-619.
- [17] Pedersen, T. & Bruce, R. (1997) Distinguishing Word Senses in Untagged Text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. Providence, RI, 197-207.
- [18] Poesio, M. & Almuhareb, A. (2005) Identifying Concept Attributes Using a Classifier. In *Proceedings of ACL Workshop on Deep Lexical Acquisition*. Ann Arbor, USA.
- [19] Poesio, M., Ishikawa, T., Walde, S. & Vieira, R. (2002) Acquiring lexical knowledge for anaphora resolution. In *Proceedings of LREC*. Las Palmas.
- [20] Purandare, A. & Pedersen, T. (2004) Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*. Boston, MA.
- [21] Pustejovsky, J. (1995) *The generative lexicon*, MIT Press.
- [22] Rapp, R. (2004) Utilizing the One-Sense-per-Discourse Constraint for Fully Unsupervised Word Sense Induction and Disambiguation. In *Proceedings of the fourth international conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal.
- [23] Schütze, H. (1992) Dimensions of meaning. In *Proceedings of Supercomputing '92*. Minneapolis, 787-796.
- [24] Schütze, H. (1998) Automatic Word Sense Discrimination. *Computational Linguistics*, 24, 97-123.
- [25] Vinson, D. P., Vigliocco, G., Cappa, S. & Siri, S. (2003) The Breakdown of Semantic Knowledge: Insights from a Statistical Model of Meaning Representation. *Brain and Language*, 86(3), 347-365.
- [26] Woods, W. A. (1975) What's in a link: Foundations for semantic networks. IN BOBROW, D. G. & COLLINS, A. (Eds.) *Representation and understanding: studies in cognitive science*. New York, Academic Press.

Integrating Domain and Paradigmatic Similarity for Unsupervised Sense Tagging

Roberto Basili¹ and Marco Cammisa¹ and Alfio Massimiliano Gliozzo²

Abstract. An unsupervised methodology for Word Sense Disambiguation, called Dynamic Domain Sense Tagging, is presented. It relies on the convergence of two very well known unsupervised approaches (i.e. Domain Driven Disambiguation and Conceptual Density). For each target word a domain is dynamically modeled by expanding the its topical context, i.e. a set of words evoking the underlying/implicit domain where the word is located. The estimation of the paradigmatic similarity within such a specific lexicon is assumed as a disambiguation model. The Conceptual Density measure is here used to account for paradigmatic associations, and the top scored senses of the target word are selected accordingly. Results confirm the impact of domain based representation in capturing useful paradigmatic generalizations, especially when small text fragments are available. In addition, the precision/recall tradeoff of the resulting method can be tuned in a meaningful way, allowing us to achieve impressively high precision scores in a purely unsupervised setting.

1 Introduction

Semantic disambiguation is a relevant problem in a number of Artificial Intelligence applications, ranging from Ontology Engineering to Information Retrieval. Any text-driven ontology population or ontology mapping task, for instance, is implicitly a sense disambiguation problem at various degrees of grain and complexity. Another important application is query translation and expansion for cross-language Information Retrieval.

Unfortunately, Word Sense Disambiguation (WSD) is a cornerstone in Natural Language Processing. In fact, the specificity of the target application domains and the involvement of languages different from English do not allow to rely on existing sense tagged resources, without whom state-of-the-art supervised approaches to WSD [6] cannot be adopted. Unsupervised approaches for semantic disambiguation thus play a central role in most of the above scenarios.

Among those approaches, [9] proposes a method to estimate predominant senses via a large untagged corpus, reaching high accuracy on standard WSD benchmarks. The underlying assumption is that predominant senses are not stable across domains and corpora, and that the notion of predominance can be modeled through an unsupervised corpus based analysis. The main limitation of that approach is that the sense distribution is modeled once for all on the basis of the corpus adopted for unsupervised learning, while word senses are first of all disambiguated by local constraints. The concept of predomi-

nance provide only an "a-priory" information about the probability distribution of word senses, that should be anyway refined by looking at a local context.

A possibility to merge both approaches is then to adapt the global information provided by the whole corpus to the local constraints provided by tokens in the surrounding context of the word to be disambiguated. Each sentence in fact suggests a topical context where the semantics of the target word is highly constrained. The modeling of such local domain evidence can thus exploit several sources of semantic evidences. Words w describing a topical contexts of the target word tw can be used to impose tighter constraints to the interpretation of tw . They express indeed a variety of information:

- *Domain information*, as the semantic domain of the context can be used to discriminate among metaphorical sense shifting (e.g. *virus* in Biology vs. Computer science)
- *Paradigmatic information*, as important siblings or synonyms w of tw (e.g. *bugs* vs. *bacteria*) are likely to appear in topical-contexts: class information for sense distinctions is thus made available

In this paper we investigate a disambiguation model that captures the above idea within an unsupervised framework aiming to address the semantic disambiguation problem of classical AI tasks, e.g. query translation and ontology learning. The model, named Dynamic Domain Semantic Tagging (*DDST*), relies on unsupervised learning techniques and Wordnet [11].

The DDST algorithm integrates two well known methodologies, i.e. the Domain Driven Disambiguation (*DDD*) [8] and the similarity metric known as Conceptual Density (*CD*) [1], reflected in the following two steps:

Domain Detection Extract the topical-context for the target word.

Paradigmatic Disambiguation Estimate the paradigmatic cohesion of the domain lexicon and then select the sense maximizing such an association within the topical context .

Details of each individual step will be provided in Section 2 and Section 3, respectively.

The DDST approach solves the main limitations of the CD algorithm: the lack of semantic relations among concepts having different ontological types (e.g. the nouns *athlete* and *sport* are totally unrelated in WordNet, because they belong to the disjointed hierarchies of *life forms* and *act*, respectively). To this aim, the DDST algorithm identifies a wider context (i.e the semantic domain) for the word to be disambiguated. In this way a wider region of WordNet is involved, providing the CD algorithm of a more reliable evidence.

The DDST algorithm is then totally unsupervised, because it does not rely on the availability of sense tagged data and it does not exploit neither a prior estimation of sense frequency distribution [9] nor the

¹ University of Rome Tor Vergata, Italy, email: {basili,cammisa}@info.uniroma2.it,

² ITC-irst, Trento, Italy, email: gliozzo@itc.it

sense ordering provided by WordNet (reflecting the sense predominance in Semcor). It can be thus applied to a large set of tasks and languages once a WordNet like lexical database and a large corpus have been provided. It represents a technologically viable solution, uniform across different languages and application scenarios.

The result of the Domain Detection step is a sensible improvement in the accuracy of the overall lexical disambiguation process. To show this, in Section 4 we compared the DDST performances against a classical CD algorithm over the Senseval-3 English all words task [10]. First, the DDST algorithm achieves a substantial improvement against a random baseline. More importantly, its superiority is particularly evident when small contexts of the target word are taken into account, and this confirms its applicability to query disambiguation (expansion or translation). Conclusion and future work are finally discussed in Section 5

2 Dynamic Generation of Domain Lexicons

The Domain Detection step is inspired by the Domain Driven Disambiguation (DDD) methodology for WSD, originally proposed by [8]. DDD is an unsupervised approach to WSD consisting on selecting the word sense that maximizes the similarity with the domain of the context. In its original formulation the DDD methodology required (i) a knowledge base containing domain information for each sense of the word to be disambiguated and (ii) an domain relevance estimation algorithm to recognize the domain of the context in which the word occurs. To accomplish the first requirement authors proposed the exploitation of WordNet Domains [7], an extension of WordNet, where each synset has been manually annotated by domain labels selected from a predefined domain set. Unfortunately, this approach is very restrictive for the following reasons:

1. The manual development of lexical resources is very expensive³.
2. The domain set is fixed once for all in the resource, while the relevant domain distinctions are usually task and context dependent.
3. Domain information alone is not sufficient to resolve lexical ambiguity: syntagmatic and paradigmatic aspects of sense distinction are not taken into account

The Domain Detection step is an attempt to solve both the first and the second problems reported above, defining a dynamic approach to identify only those domain distinctions actually involved in the application domain without requiring any manual domain annotation. In fact, in the perspective proposed by the authors, the domain distinctions have been conceived as a "static" notion, reflecting the lexicographers' intuition of an a-priori organization of the world. On the other hand, we believe that such a notion is strongly dependent on the specificity of the particular discourse or inference we are dealing with, preventing us to define such a static domain model.

For these reasons we adopted the more flexible notion of topical-context to deal with the domain modeling problem. Topical context are collections of words, describing the domain in which the text is located. They can be determined in a totally unsupervised way by adopting term/text similarity techniques, and can be used to plug "external knowledge" from a large corpus into the disambiguation process.

Such a similarity metric can be provided by defining a Domain Space [5], where both terms and texts are represented by means of Domain Vectors (DVs). We exploit the duality property of the domain space to perform the Domain Detection step: word contexts are represented by text vectors in the domain space, then the more

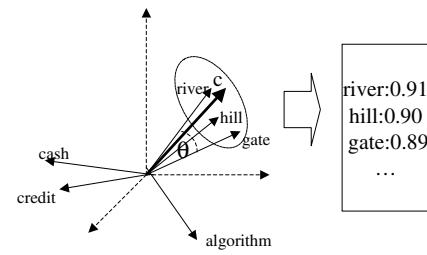


Figure 1. Output of the domain discovery step for the sentence $c = \text{"He took a walk along the bank of the river"}$

closely related terms are retrieved by exploiting the cosine similarity metric.

To show this, we illustrate the Domain Detection process by an example. Suppose that the word *bank* has to be disambiguated into the sentence "*He took a walk along the bank of the river*". The topical-context for this occurrence of *bank*, contains the terms *river*, *hill*, *snow* and *road*, among the others (see Figure 1).

The Domain Space can be defined once a Domain Model (DM) is available. A DM is represented by a $k \times k'$ rectangular matrix \mathbf{D} , containing the domain relevance for each term with respect to each domain, where k is the cardinality of the vocabulary, and k' is the size of the Domain Set. Once a DM has been defined by the matrix \mathbf{D} , the Domain Space is a k' dimensional space, in which both texts and terms are associated to Domain Vectors (DVs), i.e. vectors representing their domain relevances with respect to each domain. The DV \vec{w}_i for the term $w_i \in \mathcal{V}$ is the i^{th} row of \mathbf{D} , where \mathcal{V} is the vocabulary of the corpus. The similarity among DVs in the Domain Space is estimated by means of the cosine operation.

DMs can be acquired from texts in a totally unsupervised way by exploiting a lexical-coherence assumption [5]. To this aim, term clustering algorithms can be adopted: each cluster represents a Semantic Domain and the degree of association among terms and clusters, estimated by the learning algorithm, provides a domain relevance function. For our experiments we adopted a clustering strategy based on Latent Semantic Analysis (LSA), following the methodology described in [6]. This operation is done off-line, and can be efficiently performed on large corpora. To filter out noise, we included only those terms having a frequency higher than 5 in the corpus.

When a word w , located into the context $c = (w_{-h}, \dots, w_{-1}, w_0, w_{+1}, \dots, w_{+h})$, has to be disambiguated, first the DV \vec{c} of its context is estimated by

$$\vec{c} = \sum_{i=-h}^{+h} \vec{w}_i \quad (1)$$

\vec{c} defines a neighbourhood including the DVs \vec{w} of other corpus terms $w \in \mathcal{V}$, topically associated to the source text. A topical context for the \vec{c} can be thus formally defined by: $\{w \in \mathcal{V} \mid \| \vec{w} - \vec{c} \| < \theta\}$, where $\|\cdot\|$ is any valid norm in the LSA space derived from the cosine similarity and $\theta \in \mathbb{R}^+$. The topical-context includes all those terms whose distance (i.e. similarity) with the context c is below (above) a *domain specificity* threshold. In general, the lower the domain threshold θ , the higher the specificity of the returned topical-context and the smaller its size. Another restriction that can be imposed in the modeling of a proper topical-context is *polysemy*. In fact, ambiguous words inherently activate noisy generalizations during the similarity estimations due to their spurious sense(s). On the other hand, monosemous words should always refer to the unique correct sense, thus activating the "right" paradigmatic associations.

³ The development of WordNet Domains took about 2 man year.

In order to control such phenomenon, those words whose polysemy is over a predefined threshold ρ can be filter out. A formal definition for a topical-context in the disambiguation perspective is thus given by:

$$TC(c) = \{w \in \mathcal{V} \mid \| \vec{w} - \vec{c} \| < \theta \text{ and } pol(w) < \rho\} \quad (2)$$

where $pol(w)$ expresses the polysemy (i.e. number of senses) of w in WordNet.

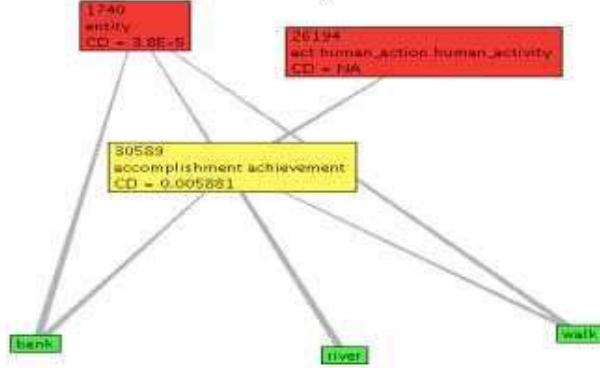


Figure 2. Conceptual Density for the words *bank*, *walk* and *river*

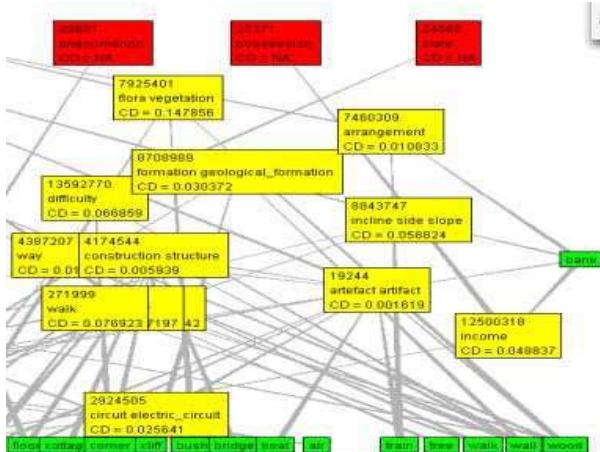


Figure 3. Conceptual Density for the word *bank* together with its topical-context

3 Semantic Disambiguation through Conceptual Density

In [2] an n -ary similarity measure aimed to support an unsupervised approach to semantic tagging has been presented. It represents a variant of the notion of *Conceptual Density* previously suggested as a tool for sense disambiguation [1]. In [3] it is applied for learning sense preferences in Wordnet from available syntagmatic patterns found in a corpus. Sets r of syntactically similar words are clustered and the best senses α as useful semantic explanations of the underlying syntactic phenomena are selected. In a Bayesian perspective, rules for disambiguation (i.e. estimates for $prob(\alpha|r, w)$) are derived from similar syntactic contexts r , i.e. the sets of different words w in the same grammatical contexts. In order to favour some senses an estimation of semantic similarity "local" to the contexts r is carried out via the *conceptual density* ($cd^r(\alpha)$) measure.

The same definition of conceptual density as in [3] is proposed here for the semantic disambiguation phase. However, a different notion of context r is applied. Instead of using syntactic equivalence

(i.e. occurrence in the same syntactic contexts), we define r as the *neighbourhood* $TC(c)$ determined by Eq. 2 for the source context c . The hypothesis is that some members in a topical context $TC(c)$ are expected to share paradigmatic properties with the target word tw : their sense in $TC(c)$ is thus semantically close to the suitable sense of the tw in the source context c . The closest sense α is thus selected by maximizing the conceptual density of the Wordnet hierarchy rooted at α .

As illustrated by Figure 3, the topical-context provide more evidence than the mere list of the words actually located in context of the word to be disambiguated (i.e. *walk* and *river*, see Figure 2) improving the final disambiguation. In fact, the (wrong) generalization for the word *bank* provided by the classical CD algorithm is *accomplishment achievement*, while if the topical-context is taken into account the CD algorithm select the (correct) sense of *incline slope land*.

3.1 A similarity measure for large scale semantic disambiguation

In Section 2 a topical context $TC(c)$, i.e. a domain specific lexicon, originating from a textual context c has been defined via Eq. 2. Words $w \in T(c)$ can be generalized through senses α of the Wordnet hierarchy. The likelihood of a sense α_{tw} for the target word tw is proportional to the number of other words $w \in TC(c)$ that have common generalizations with tw along the paths activated by their α in the hierarchy. A measure of the suitability of senses α for words $w \in TC(c)$ is thus the *information density* of the subtrees rooted at α . The higher is the number of nodes in the subhierarchy that generalize some nouns $w \in TC(c)$, the better is the related interpretation α . The conceptual density models the former notion and provides a measure for the latter.

DEF (Conceptual Density). Given a context c for a target word tw , its topical context $TC(c)$, a synset α in Wordnet used to generalize n different nouns $w \in TC(c)$, the conceptual density, $cd^{TC(c)}(\alpha)$, of α with respect to $TC(c)$ is defined as:

$$cd^{TC(c)}(\alpha) = \frac{\sum_{i=0}^h \mu^i}{area(\alpha)} \quad (3)$$

where :

- h is the estimation of the depth of a tree able to generalize the n nouns. Its actual value is estimated by:

$$h = \begin{cases} \lfloor \log_\mu n \rfloor & \text{if } \mu \neq 1 \\ n & \text{otherwise} \end{cases} \quad (4)$$

- μ is the average number of sons per node in the full Wordnet subhierarchy dominated by α . Its estimation is available statically from Wordnet and can be evaluated *a priori* without uncertainty. Notice that when nodes belong to unbalanced branches of the hierarchy, the value for μ can approach (and in fact is) 1, so that a specific treatment of them is needed in the definition.
- $area(\alpha)$ is the number of nodes in the α subhierarchy. This value is also estimated statically from Wordnet.

Equation 3 applies to *any* valid and common generalization of the nouns $w \in TC(c)$ as topically related to tw . The aim of this method however is to reduce as much as possible the number of such generalizations. In [3] two properties have been defined that determine the best generalization set $G(c)$ among all the valid generalization set S applicable to an originating context c , i.e.

$$S = \{\alpha | \alpha \text{ is an hyperonim of at least one sense } \sigma_w, w \in TC(c)\}.$$

a) (*Useful generalization*) S is a useful set of generalizations if S covers the entire set $TC(c)$ and $\forall \alpha \in S$, α is an hyperonym of at least two words $w_1 \neq w_2 \in TC(c)$

b) (*Maximally dense*) The factor $\sum_{\alpha \in S} cd^{TC(c)}(\alpha)$ must be maximal, among the different S in the family of useful sets of generalizations, i.e.

$$G(c) = \operatorname{argmax}_S \sum_{\alpha \in S} cd^{TC(c)}(\alpha)$$

where S satisfies a).

The $G(c)$ set characterized by a) and b) is the *best paradigmatic interpretation* of the topical context $TC(c)$ generated by c . In [3] a *greedy* algorithm able to efficiently compute the minimal set $G(c)$ that covers $TC(c)$ with a maximal density is defined⁴. The outcome is the minimal set $G(c)$ of synsets that are the *maximally dense generalizations of at least two words* in $TC(c)$. Words $w \in TC(c)$ for which no such generalization may exist and will not be represented in the resulting set $G(c)$. Scores $cd^{TC(c)}(\alpha)$ model the strength by which words $w \in TC(c)$ are related to senses α , as suitable *paradigmatic explanations* of the topical context $TC(c)$.

The semantic disambiguation of a target word tw in a context c depends on the subset of generalizations $\alpha \in G(c)$ concerning some of its senses σ_{tw} . Let $G_{tw}(c)$ be such a subset, i.e.

$$G_{tw}(c) = \{\alpha \in G(c) \mid \exists \sigma_{tw} \text{ such that } \sigma_{tw} \prec \alpha\} \quad (5)$$

where \prec denotes the transitive closure of the hyponymy relation in Wordnet.

DEF (Semantic disambiguation). Given a context c for a target word tw and the generalization set $G_{tw}(c)$ defined above, the set $\sigma(tw, c)$ of correct sense(s) for tw in c is given by:

$$\sigma(tw, c) = \{\sigma_{tw} \mid \sigma_{tw} \prec \bar{\alpha}\} \quad (6)$$

with $\bar{\alpha} = \operatorname{argmax}_{\alpha \in G_{tw}(c)} cd^{TC(c)}(\alpha)$. Notice that when $G_{tw}(c)$ is empty then also $\sigma(tw, c) = \emptyset$ so that no decision can then be taken for the underlying context c . Other disambiguation heuristics should be applied. Moreover, it is also possible that $|\sigma(tw, c)| > 1$ so that multiple senses may be assigned to a context c . Equation 6 thus defines an (unsupervised) semantic disambiguation model. It will be evaluated in the next section.

4 Evaluation

In order to show the benefits obtained from adding the topical-context words, we contrasted the DDST with a classical CD algorithm over a WSD benchmark. This evaluation helps in the objective assessment of the method, but it focuses on just one of the variety of tasks to which DDST applies.

In this section we first describe the details of the implementation of the DDST algorithm and the adopted WSD benchmark (Subsection 4.1). Then, we contrast the performances of the DDST algorithm with those of a classical CD approach to WSD (Subsection 4.2). Finally, Subsection 4.3 investigates the precision/recall trade-off.

4.1 Experimental Settings

For the experiments, we used a C++ implementation of the CD algorithm described in Subsection 3, developed at the University of Rome

⁴ A demo version of the algorithm and the conceptual density measure can be accessed at <http://ai-nlp.info.uniroma2.it/Estimator/cd.htm>

Tor Vergata. Independently from the target task, the input of the tool is a set of nouns and the output is a set of generalization synsets from WordNet 1.7.1 with the related CD values. For the WSD purposes, we concentrated just on the CD estimation of the target words. The tool also returns a probability estimation (based on the CD score) for each sense of the input words. In the experiments The sense of the target word with maximal probability is selected as a final output. The DM has been acquired from the British National Corpus [4], following the methodology described in [6].

Results reported in the following subsections have been evaluated on the Senseval-3 English All-Words task [10]. This task consists on tagging all the words contained in tree middle sized English texts (i.e about 1500 words each). As the CD algorithm depends on the topology of the lexical hierarchy, DDST applies best to nouns. Then, for our experiments, we concentrated on the subset of the target words made by the 876 nouns. The Senseval tasks also provides an automatic scoring procedure for *precision* and *recall*. As a baseline for the task, we implemented a random WSD algorithm, by assigning a randomly chosen sense to each target word, achieving $F1=0.39^5$. One peculiarity of the Senseval-3 task is that wide contexts are made available to the WSD algorithm. However, in many practical applications (e.g. the disambiguation of IR queries) this assumption is unrealistic, therefore in this paper we will try to minimize the context window to be taken into account for disambiguation as much as possible.

A preliminary investigation about the role of the underlying parameters has been carried. No significant variations are observed⁶. The following setting is thus adopted for the dimensionality, similarity and polysemy control, respectively: $k' = 100$, $\theta = 0.5$, $\rho = 2$.

4.2 Role of the Topical Context

To demonstrate the usefulness of the Domain Detection step, and the effectiveness of the proposed methodology, we compared the DDST results to those of a classical CD algorithm on the Senseval-3 task. Results are reported in Figure 4, where the two algorithms have been compared by varying the context size (i.e. $h = 5, 10, 20, 50$). DDST is largely above the random baseline for every test. DDST and CD converges asymptotically when the context size increase, while the former is always superior in our experiments. Improvements are impressive especially when small contexts are considered, increasing the accuracy from 29,45 to 52,05 when a window of ± 5 tokens is considered. Surprisingly, the performances of DDST slightly decrease when wider contexts are considered, in contrast to CD. However, data show that topical contexts generated from smaller text fragments have performance sensibly higher than larger text portions (52% vs. 48%).

In general, all the experiments confirm that the implicit domain of a text is a fundamental aspect to capture lexical semantics. This aspect is better addressed by adopting unsupervised learning techniques to expand small contexts with topical-contexts rather than considering larger fragments of the originating document. The role of the topical contexts is then crucial for all those application scenarios where just small texts are available, as in most of the typical IA settings (e.g. dialogue systems, natural language interfaces).

⁵ $F1$ is the harmonic mean between precision and recall, i.e. $F1 = \frac{2pr}{p+r}$

⁶ The difference between the $F1$ achieved by the best configuration and those achieved by the worst is about 0.03.

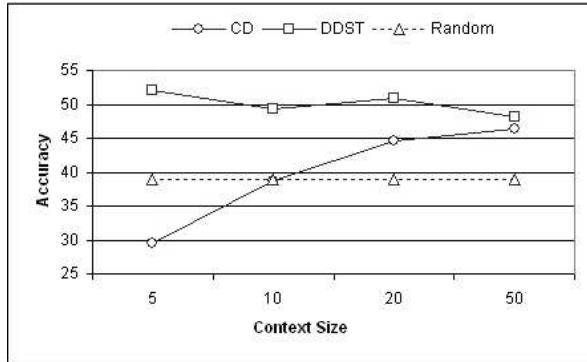


Figure 4. Accuracy varying the context size

4.3 Maximizing Precision

A relevant aspect to be taken into account when performing WSD is *precision*. In many cases, it would be preferable to tag only those words on which the algorithm is more confident, and reduce the tagging noise. For example, when WSD is used for query expansion, the wrong sense assignments would turn out in wrong expanding words, that result in misleading queries.

The DDST algorithm allows to define an effective selection criterion as its output can be considered as a probability distribution over the senses of the target word. This is done by normalizing the cumulative CD values of individual senses⁷ by the whole contribution. As this normalization gives rise to a probability estimate, then the entropy ($E(tw) = -\sum_{i=1}^n p_i(tw) \cdot \log(p_i(tw))$) with p_i as the probability of the i -th sense being correct for tw , and the perplexity, i.e. its exponential (i.e. $pp(tw) = 2^{E(tw)}$) can be measured. In particular the perplexity can be considered as an estimation of the confidence of the prediction provided by the DDST algorithm. The adopted selection criterion can thus be obtained by thresholding the perplexity of the output probability by a parameter τ .

Figure 5 reports Precision, Recall, F1 and Coverage of the system for different values of τ , showing that this parameter can be tuned meaningfully to control the trade-off between precision and recall. As shown in Fig. 5, τ allows to capture accurate predictions. Precision is above 80% along with reasonable recall values ($\tau \sim 0.1$), while a coverage of about the 60% of the cases can be still achieved with a 70% of precision ($\tau \sim 1.8$).

5 Conclusion and Future Work

In this paper we proposed a fully unsupervised methodology for semantic disambiguation useful in a large set of application scenarios, such as cross language information retrieval, question answering and ontology learning. The overall model combines in a natural and effective way distributional information observable in a corpus (mainly domain-specific topical associations among words) with the topological (i.e. paradigmatic) properties of a semantic network. A relevant side-effect of the disambiguation step is capturing paradigmatic properties, projecting out of the hierarchy a meaningful subportion as a suitable interpretation local to the source context. Such dynamic "view" can be seen as a domain-specific ontology that better characterizes the target text fragment. Results clearly shows that our ap-

proach works very well when small contexts of the target words are taken into account (e.g. queries and glosses), and can be tuned in order to boost precision to very high scores (over 80%). This is a very important feature in many IA settings, such us ontology population. Although experiments have been run over a classical WSD task, they demonstrate the technological perspectives opened by the proposed method over typical complex tasks in modern AI. This paper is just a first attempt to provide a dynamic notion to the concept of semantic domain and its relation to the paradigmatic structure in WordNet. For the future, we plan to further extend this methodology to address several lacks of this research. In particular we plan to improve the CD algorithm by providing a corpus score to each synset and to integrate syntagmatic constraints in the disambiguation process. In addition we plan to exploit the proposed technique to address the traditional AI problem tuning and large scale ontology to a specific domain.

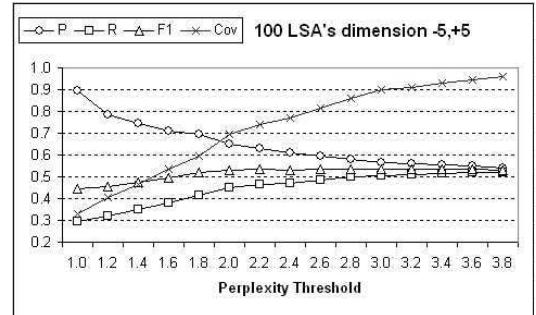


Figure 5. Precision, Recall, F1-measure, Coverage for 100x100 LSA matrix with ± 5 context at various thresholds τ

REFERENCES

- [1] E. Agirre and G. Rigau, 'Word sense disambiguation using conceptual density', in *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 16–22, Copenhagen, Denmark, (1996).
- [2] R. Basili and M. Cammisa, 'Unsupervised semantic disambiguation.', in *Proceedings of the LREC 04 Workshop Beyond Named Entity Recognition: Semantic labelling for NLP tasks*, Lisbon, (2004).
- [3] Roberto Basili, Marco Cammisa, and Fabio Massimo Zanzotto, 'A semantic similarity measure for unsupervised semantic disambiguation', in *Proceedings of the Language, Resources and Evaluation LREC 2004 Conference*, Lisbon, Portugal, (2004).
- [4] BNC-Consortium. British national corpus., 2000.
- [5] A. Gliozzo, *Semantic Domains in Computational Linguistics*, Ph.D. dissertation, ITC-irst/University of Trento, 2005.
- [6] A. Gliozzo, C. Giuliano, and C. Strapparava, 'Domain kernels for word sense disambiguation', in *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL-05)*, pp. 403–410, Ann Arbor, Michigan, (June 2005).
- [7] B. Magnini and G. Cavaglià, 'Integrating subject field codes into WordNet', in *Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation*, pp. 1413–1418, Athens, Greece, (June 2000).
- [8] B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo, 'The role of domain information in word sense disambiguation', *Natural Language Engineering*, 8(4), 359–373, (2002).
- [9] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, 'Finding predominant senses in untagged text', in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 280–287, Barcelona, Spain, (2004).
- [10] R. Mihalcea and P. Edmonds, eds. *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July 2004.
- [11] G. Miller, 'An on-line lexical database', *International Journal of Lexicography*, 13(4), 235–312, (1990).

⁷ Different generalizations in $G_{tw}(c)$ may cover the same sense $\sigma(tw, c)$ providing different evidence in favour of $\sigma(tw, c)$.

Disambiguating Personal Names on the Web using Automatically Extracted Key Phrases

Danushka Bollegala¹ and Yutaka Matsuo² and Mitsuru Ishizuka³

Abstract. When you search for information regarding a particular person on the web, a search engine returns many pages. Some of these pages may be for people with the same name. How can we disambiguate these different people with the same name? This paper presents an unsupervised algorithm which produces unique phrases to disambiguate different people with the same name (i.e. namesakes). Our algorithm takes in a personal name and outputs multiple sets of phrases which uniquely identify the different namesakes on the web. These phrases could then be added to the query to narrow down the search to a specific namesake. We evaluated the algorithm on a collection of documents retrieved from the Web. Experimental results show a significant improvement over the existing methods proposed for this task.

1 Introduction

The Internet has grown into a collection of billions of web pages. One of the most important interfaces to this vast information are web search engines. We send simple text queries to search engines and retrieve web pages. However, due to the ambiguities in the queries and the documents, search engines return lots of irrelevant pages. In the case of personal names, we may receive web pages to other people with the same name (*namesakes*). However, the different namesakes appear in quite different contexts. For example if we search for *Michael Jackson* in Google, among the top hundred hits we get a beer expert and a gun dealer along with the famous singer. Although namesakes share a common name, in most of the cases they appear in entirely different contexts. For example, in the case of Michael Jackson, terms such as *music*, *album*, *trial* associate with the famous singer, whereas we observe terms *beer*, *travel*, *hunter* for the *beer expert* namesake. This paper proposes an unsupervised algorithm which extracts such uniquely identifying key phrases from the Web to disambiguate people with the same name. These automatically extracted key phrases could then be used to modify the original query and narrow down the search.

Disambiguating namesakes on the Web is a challenging task. To begin with, the number of namesakes for a particular name is unknown. Moreover, not all namesakes are equally represented on the Web. In many cases there are two or three famous namesakes which have lots of pages about them and all other namesakes have just one or two pages on them. Another difficulty is identifying the scope of the context of a name. An entire web page/site may be written on a person (for example home pages and fan sites) or the name may ap-

pear on passing (for example book reviews on Amazon mentioning an author of a book, conference programs mentioning names of the authors of papers, etc). On the other hand there are cases where an individual has various web appearances. For example the renowned linguist Noam Chomsky appears as a linguist and also as a critic of American foreign policy. It would be interesting to see how a namesake disambiguation method responds to such complications.

Disambiguating namesakes is vital for social network extraction systems [14, 17]. Matsuo et al [14], proposes a social network extraction system in which they measure the connection between two persons *A* and *B* based on the number of hits for the query *A AND B* on a web search engine. However, this method cannot be used to create social networks for namesakes because of the ambiguity of the names. We can easily overcome this limitation by including a phrase in the query, that uniquely identifies the person under consideration from his or her namesakes.

2 Related Work

There is little previous work we know of that directly addresses the problem of extracting key phrases to disambiguate personal names on the web, but some related problems have been studied. Disambiguation of namesakes is similar to tuple matching in databases—the problem of deciding whether multiple relational tuples from heterogeneous sources refer to the same real-world entity [8, 1, 9].

From a natural language perspective, there has been a lot of work on the related problem of co-reference resolution [2, 15]. The goal in co-reference resolution is to link occurrences of noun phrases and pronouns, typically occurring in a close proximity, within a few sentences or a paragraph, based on their appearance and local context. Co-reference resolution is vital for many natural language tasks such as text summarization, question answering and entity extraction [5]. Various algorithms have been proposed for co-reference resolution. Fundamentally, these algorithms map the local information around a pronoun to a set of features and use a machine learning technique to determine whether a given pronoun corresponds to a given noun phrase.

A few works address the problem of personal name disambiguation across a collection of documents. Mann, et al [13] considers the problem of distinguishing occurrences of a personal name in different documents. They propose an unsupervised algorithm which extracts *people-specific* biographical information such as birth date, birth place, occupation etc using a set of regular expressions to cluster the documents to their namesakes. However, such person-specific information is not always available for all the namesakes on the web. Even in cases where such information is available, a set of fixed regular expressions as used by Mann et al [13] is not sufficient to ex-

¹ University of Tokyo, danushka@mi.ci.u-tokyo.ac.jp

² Japanese National Institute of Advanced Industrial Science and Technology, y.matsuo@aist.go.jp

³ University of Tokyo, ishizuka@i.u-tokyo.ac.jp

tract them. Bekkerman, et al [4] proposes a link structure model and an agglomerative-conglomerative double clustering (A/CDC) based algorithm to disambiguate a group of people on the web. The algorithm assumes our ability to obtain information regarding the social network (associates) of the person to be disambiguated. The method can be readily used when we have such information. However, in most of the situations we do not know well enough about the associates of the person which we want to disambiguate. Pedersen et. al. [18] proposes a method for discriminating names by clustering the instances of a given name into groups. They extract the context of each instance of the ambiguous name and generate second order context vectors using significant bigrams. The vectors are clustered such that instances that are similar to each other are placed into same clusters. Li, et al [12] suggests an algorithm which could be used to disambiguate not only personal names but other types of named entities such as organizations and locations. They propose a discriminative model based on agglomerative clustering and a generative model which uses a language model combined with EM algorithm. Their experimental results show that the generative model out performs the discriminative model. They do not discuss the topic of extracting key phrases to distinguish the different entities.

3 Method

3.1 Problem Settings and Modeling

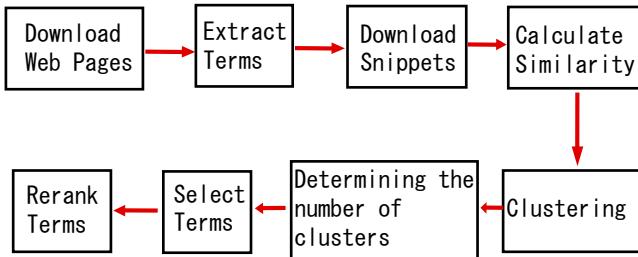


Figure 1. Outline of the method

The outline of our method is illustrated in figure 1. Our method takes the name to be disambiguated as the input and outputs a list of key phrases for each of the different namesakes. The first step is to collect a set of documents that covers all the namesakes for a given name. This step could be omitted in cases where a particular name is to be disambiguated in a given document collection. Collecting a set of documents from the Web that covers all namesakes of a given name is beyond the scope of this paper. In our system, to collect information regarding a particular name we use Google⁴ and download the top 100 web pages for the given name.

We assume each document in the collection to represent exactly one namesake. This assumption allows us to identify each document in the collection with a particular namesake. Thus, the problem of disambiguation becomes a one of document clustering. We cluster the set of documents such that each cluster represent a different namesake, and extract key phrases from each cluster to identify the namesake it represents.

⁴ <http://www.google.com/apis>

3.2 Term Model

Contextual Hypothesis for Senses [21] states that two occurrences of an ambiguous word belong to the same sense to the extent that their contextual representations are similar. According to this hypothesis, if the contexts of which two instances of a name appears are similar, then we could infer that both the instances of the name belong to the same person. In order to achieve this goal, we need a model that represents all the salient knowledge regarding a particular instance of a name. Traditionally, a document is modeled as a bag-of-words and represented by a word vector [19]. However, in our case the context of an instance of a name may vary from few lines to an entire web page. Considering all the words (excluding stop words) adds too much noise. Moreover, a document may not totally focus on the namesake, but also contain lots of other information. The knowledge representation model should be robust enough to capture the salient information for disambiguation.

In this paper, we propose *Term Models* as our knowledge representation model. A Term Model $T(A) = t_1, \dots, t_N$, of a personal name A (since we consider each document as representing exactly one namesake, we have exactly one term model for each document) is defined as the set of N terms t_i, \dots, t_N extracted from the context of a personal name. In our algorithm we consider the context of a name in a document to be the entire document and extract terms from the entire document. We use *C-value* [6, 7], an automatic term recognition algorithm, to extract multi-word terms.

The *C-value* approach combines linguistic and statistical information. The linguistic information consists of the part-of-speech tagging of the document being processed, the linguistic filter constraining the type of terms extracted and the stop lists. The statistical part combines statistical features of the candidate string. The linguistic filter contains a predefined set of patterns of nouns, adjectives and prepositions that are likely to be terms. The stop list is a list of words which are not expected to occur as term words in a given domain. The combinations of nouns, adjectives and prepositions that are allowed by the linguistic filter and the stop list are considered as the potential candidates as terms. The *termhood* (likeliness of a candidate to be a term) is evaluated using C-value. C-value is built using statistical characteristics of the candidate string, such as, the total frequency of occurrence of the candidate string in the document, the frequency of the candidate string as part of other longer candidate strings, the number of these longer candidate terms and the length of the candidate string (in number of words). C-value is defined as follows,

$$C\text{-}value(a) = \begin{cases} \log_2 |a| \cdot f(a) & \text{ais not nested,} \\ \log_2 |a|(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases} \quad (1)$$

where, a is the candidate string, $f(a)$ is its frequency of occurrence in the document, $|a|$ is the length of the candidate string, T_a is the set of extracted candidate terms that contain a , $P(T_a)$ is the number of candidate terms.

3.3 Similarity Calculation

In order to cluster the documents using the term model explained in previous section, we need to calculate the similarity between two documents. Exact matches of terms are rare. Therefore, we would require a similarity metric which is capable of comparing the terms at a semantic level. For example, the two terms *George Bush* and

“George Bush”	“The president of the United States”
(1) Official White House site presents issue positions, news, Cabinet, appointments, offices and major speeches. Includes biography, video tour and photo ...	(1) Whitehouse.gov is the official web site for the White House and President George W. Bush, the 43rd President of the United States.
(2) Official Internet home of the Republican National Committee. Updated daily with news and commentary from the RNC.	(2) Background information, election results, cabinet members, notable events, and some points of interest on each of the presidents.
(3) George Bush (41st President: 1988–1992).	(3) For a list of persons who served as the President of the United States following the ratification of the United States Constitution see the list of ...
(4) Zack Exley’s satirical site of George W. Bush campaign, now quite overt.	(4) A history of presidents, the presidency, politics and related subjects. Includes biographies for every president.
(5) Parody of official White House web site. Includes spoof news and gossip.	(5) ... Presidents of the Continental Congress as well as information about David Rice Atchison who some believe was the 12th President of the United States. ...

Figure 2. Top five snippets extracted for two terms

The *president of the United States* are closely related but do not have any words in common. Word Net⁵ based similarity metrics have been widely used as semantic similarity measures between words in sense disambiguation tasks [16, 3]. However, personal names are not covered in the Word Net. Sahami et al [20] proposes a method to calculate similarity between terms using snippets retrieved by a web search engine. A Snippet is a small piece of text, containing two or three sentences extracted from the document around the query term. Most web search engines provide snippets as short summaries of the search results.

For example, consider the first five snippets returned by Google for *George Bush* and *The president of the United States* in figure 2. Even among the first five snippets for these two terms, we find many common terms such as *President*, *White House*, *Official*, *and*, *site*, etc. For a given term we collect its snippets and construct the distribution of words in the snippet. The frequency of each word in the collection of snippets is divided by the total number of words. We compute Kullback-Liebler (KL) divergence, as a measure of dissimilarity of the two terms. For two probability distributions $p(x)$ and $q(x)$, which are defined over a random variable $x \in X$, their KL-divergence $D(p||q)$ is defined as follows,

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \quad (2)$$

Therein, X is the vocabulary. KL-divergence becomes undefined when there are words with zero probabilities. Skew divergence is used to overcome this problem [11]. Skew divergence $S_\alpha(p, q)$ is defined as follows,

$$S_\alpha(p, q) = D(q||\alpha p + (1 - \alpha)q). \quad (3)$$

Therein: $\alpha \in [0, 1]$ is the degree of skewness between the two distributions p and q . In order to transform the asymmetrical skew diver-

⁵ <http://wordnet.princeton.edu/perl/webwn>

gence to symmetric similarity measure $\text{sim}(p, q)$, we take the average divergence as follows,

$$\text{sim}(p, q) = \exp(-\frac{1}{2}(S_\alpha(p, q) + S_\alpha(q, p))). \quad (4)$$

In our implementation, we considered the top 100 snippets from Google and set $\alpha = 0.99$.

Let $T(A) = \{a_1, \dots, a_n\}$ be the term model of document A and $T(B) = \{b_1, \dots, b_m\}$ the term model of document B . Here, a_1, \dots, a_n are n terms extracted from document A and b_1, \dots, b_m are m terms extracted from document B . We define the similarity, $\text{DocSim}(A, B)$, between two documents A and B using their term models $T(A), T(B)$ as follows,

$$\text{DocSim}(A, B) = \frac{1}{mn} \sum_{a_i \in A; b_j \in B} \text{sim}(a_i, b_j). \quad (5)$$

Here, $\text{sim}(a_i, b_j)$ is calculated using equation 4 based on the probability distributions.

3.4 Clustering

We use group-average agglomerative clustering (GAAC) to cluster the documents to their namesakes. Initially, each document is assigned to a separate cluster. GAAC in each iteration executes the merger that gives rise to the cluster Γ with the largest average correlation $C(\Gamma)$ where,

$$C(\Gamma) = \begin{cases} \frac{1}{2} \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{u \in \Gamma} \sum_{v \in \Gamma} \text{DocSim}(u, v) & |\Gamma| = 1, \\ \text{otherwise.} & \end{cases} \quad (6)$$

Therein: $|\Gamma|$ denotes the number of documents in the merged cluster Γ ; u and v are two documents in Γ and $\text{DocSim}(u, v)$ is given by equation 5.

3.5 Cluster Quality

Ideally, clustering process should terminate when the number of formed clusters matches the number of different namesakes in the document collection. However, in real world problems the number of namesakes for a given name is not known. Clustering in general can be considered as an optimizing problem. In clustering we try to;

1. maximize the similarity of documents within a cluster,
2. minimize the similarity of documents between clusters.

We prefer our clusters to be well correlated internally and each of the clusters to be different among themselves. The quality (goodness) of the formed clusters can be evaluated based on how well the clusters satisfy these two conditions [10]. We define *internal correlation* as a measure of how well the first condition is satisfied (i.e. the degree of similarity of documents within clusters). Internal correlation, $\text{IntCor}(\Lambda)$, of a set Λ of n clusters c_1, c_2, \dots, c_n is defined as follows,

$$\text{IntCor}(\Lambda) = \frac{1}{n} \sum_{\Gamma \in \Lambda} C(\Gamma). \quad (7)$$

Where, $C(\Gamma)$ is the average correlation defined in equation 6. We define *external correlation* as a measure of how well the second condition is satisfied. Using the above notation, external correlation, $\text{ExtCor}(\Lambda)$, is defined as the dis-similarity between the two most similar clusters in Λ as follows,

$$\text{ExtCor}(\Lambda) = 1 - \frac{1}{|\Gamma_a||\Gamma_b|} \sum_{u \in \Gamma_a} \sum_{v \in \Gamma_b} \text{DocSim}(u, v). \quad (8)$$

Where,

$$(\Gamma_a, \Gamma_b) = \arg_{\Gamma_i, \Gamma_j \in \Lambda} \min C(\Gamma_i \oplus \Gamma_j) \quad (9)$$

and the operator \oplus denotes the merging operation between two clusters. Using equations 7 and 8 we define *Cluster Quality*, $Q(\Lambda)$, as follows,

$$Q(\Lambda) = \frac{1}{2} (\text{IntCor}(\Lambda) + \text{ExtCor}(\Lambda)). \quad (10)$$

To label the clusters we select all the terms that appear in a cluster for a certain namesake but do not appear in other clusters.

4 Results and Discussion

4.1 Test Data

We evaluated our algorithm on pseudo names as well as naturally ambiguous names. For automated pseudo name evaluation purposes, we collected 150 (50 per person) documents from the web for three different people for conflation. Our collection contains documents for *Maria Sharapova* the tennis player, *Bill Gates* chairman Microsoft and *Bill Clinton* former president of the United States. We then replace every occurrence of these names in the documents with *person-x*. We evaluate the algorithm on naturally ambiguous names such as *Jim Clark*, *William Cohen*, *Tom Mitchell* and *Michael Jackson*⁶. To evaluate our algorithm on people with different web appearances we tested for *Noam Chomsky*.

⁶ This collection contains 50 documents per ambiguous name

4.2 Disambiguation Accuracy

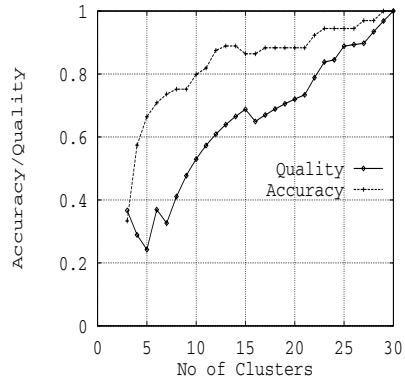
We assign each cluster to the namesake that appears most in that cluster (*holder* of the cluster). Precision, $P(C)$, of cluster C is calculated as follows,

$$P(C) = \frac{\text{No of docs in } C \text{ for its holder}}{\text{Total No of docs in } C}. \quad (11)$$

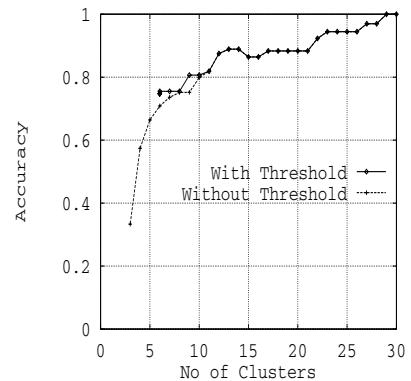
However, some namesakes have lots of documents on them, where as others are mentioned only in few documents. To reflect this fact in our evaluation metric we define *Disambiguation Accuracy*, as the weighted sum of each cluster's precision. Disambiguation accuracy (Accuracy) is defined as follows,

$$\text{Accuracy} = \sum_{C \in \Lambda} P(C) \frac{\text{No of docs in the collection for the holder of } C}{\text{Total No of docs in the collection}}. \quad (12)$$

Where, Λ is the set of clusters and $P(C)$ is given by equation 11.



(a) Accuracy/Quality Vs No of Clusters without Quality Threshold



(b) Accuracy Vs No of Clusters, with and without Quality Threshold

Figure 3. Effect of the quality threshold

Figure 3(a) depicts the accuracy/quality vs the number of clusters in the experiment with pseudo names. It shows that accuracy approximately correlates with cluster quality. This relationship enables us to

guide the clustering process based on unsupervisedly calculated cluster quality. Moreover, figure 3(a) shows that accuracy drops steadily as the number of clusters decreases. This is due to the outliers that get attached to the otherwise pure (representing the dominant namesakes) clusters. To avoid this, we terminate clustering when cluster quality drops below a fixed threshold and classify the remaining documents to the clusters. As seen from figure 3(b), this procedure prevents ill-formed clusters and yields high accuracy values. We experimentally set the cluster quality threshold to 0.6.

Table 1. Accuracy for ambiguous names

Name	Number of namesakes	Proposed	TF IDF
Jim Clark	8	71.95	59.20
Michael Jackson	3	94.96	88.76
William Cohen	10	72.71	57.96
person-X	3	81.10	39.88
Noam Chomsky	2	94.19	82.79

Table 1 shows results of our experiments. We implemented a TF-IDF based clustering algorithm as the baseline for comparison. In the baseline method, each document in the collection is represented by a word vector. We consider all the words (except a fixed list of stop words) in the document and calculate their term frequencies (TF: Term Frequency) in the document. For each word we find the number of documents containing it (DF: document frequency). Using TF,DF values we represent each document as a vector of words, with TF-IDF weights. Similarity between two documents is calculated as the cosine similarity of their word vectors. We then use GAAC to cluster the documents. Base line method utilizes the same cluster quality threshold as the proposed method. However, note that the TF-IDF based method does not produce any key phrases. Table 1 reports higher accuracy values for the proposed method over the baseline. There is a clear advantage to the proposed method over the baseline for person-X collection. Considering the fact that person-X collection contains three very different personalities, their term models are sufficiently discriminative to clearly separate the three personalities. However, for highly ambiguous names table 1 reports comparatively low accuracy values. One reason for this behavior is that, the distribution of documents among namesakes is not being even. Some of the namesakes are not sufficiently represented on the web to build a descriptive term model.

Our algorithm finds key phrases such as *racing driver Jim Clark, Formula One World Championships and motor racing* for the racing car driver-Jim Clark and *Silicon Valley, netscape* for the founder of netscape -Jim Clark. In the case of Michael Jackson, the top ranking terms for the singer are *Fan Club, World network, news, Micheal Jackson case and pop star*. The proposed method had the lowest accuracy for william cohen as it found only three out of the ten namesakes in the collection. In the person-X experiments, we find key phrases such as *first set, US open, Wimbledon, Venus Williams and Grand Slam title* for Maria Sharapova, *wealthiest person, Microsoft* for Bill Gates and *White house, former president* for Bill Clinton. Although, Noam Chomsky is not an ambiguous name, we tested on it to evaluate the algorithm on individuals with different web appearances. Interestingly, the algorithm produces key phrases such as *preventive war, government complicity, George Bush, Tony Blair* in the Chomskey the critic cluster and *universal grammar, linguistic theory* in the Chomskey the linguist cluster.

5 Conclusion

We proposed and evaluated an algorithm to extract key phrases from the web, to disambiguate personal names. The algorithm is unsupervised and uses a cluster quality metric to determine the number of namesakes. Our experiments show encouraging results. In future, we intend to explore the possibilities to extend the proposed method to disambiguate other types of named entities.

REFERENCES

- [1] P. Andritsos, R.J. Miller, and P. Tsapars, ‘Information-theoretic tools for mining database structure from large data sets’, in *Proceedings of the ACM SIGMOD Conference*, (2004).
- [2] A. Bagga and B. Baldwin, ‘Entity-based cross-document coreferencing using the vector space model’, in *Proceedings of COLING*, pp. 79–85, (1998).
- [3] Sutanjeev Banerjee and Ted Pedersen, ‘An adapted lesk algorithm for word sense disambiguation using word net’, in *Proceedings of the third international conference on computational linguistics and intelligent text processing*, pp. 136–145, (2002).
- [4] Ron Bekkerman and Andrew McCallum, ‘Disambiguating web appearances of people in a social network’, in *Proceedings of the 14th international conference on World Wide Web*, pp. 463–470, (2005).
- [5] Matthias Blume, ‘Automatic entity disambiguation: Benefits to ner, relation extraction, link analysis, and inference’, in *Proceedings of International Conference on Intelligence Analysis*, (2005).
- [6] K.T. Frantzi and S. Ananiadou, ‘Extracting nested collocations’, in *16th Conference on Computational Linguistics*, pp. 41–46, (1996).
- [7] K.T. Frantzi and S. Ananiadou, ‘The c-value/nc-value domain independent method for multi-word term extraction’, *Journal of Natural Language Processing*, **6**(3), 145–179, (1999).
- [8] M. Hernandez and S. Stolfo, ‘The merge/purge problem for large databases’, in *SIGMOD Conference*, pp. 127–138, (1995).
- [9] Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen, ‘Exploiting relationships for domain-independent data cleaning’, in *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, (April 21–23 2005).
- [10] Ravi Kannan, Santosh Vempala, and Adrian Vetta, ‘On clustering: Good, bad and spectral’, *Computer Science*, (2000).
- [11] Lillian Lee, ‘On the effectiveness of the skew divergence for statistical language analysis’, *Artificial Intelligence and Statistics*, 65–5, (2001).
- [12] Xin Li, Paul Morie, and Dan Roth, ‘Semantic integration in text, from ambiguous names to identifiable entities’, *AI Magazine, American Association for Artificial Intelligence*, Spring, 45–58, (2005).
- [13] Gideon S. Mann and David Yarowsky, ‘Unsupervised personal name disambiguation’, in *Proceedings of CoNLL-2003*, pp. 33–40, (2003).
- [14] Yutaka Matsuo, Junichiro Mori, and Masahiro Hamasaki, ‘Polyphonet: An advanced social network extraction system from the web’, in *Proceedings of the World Wide Web Conference*, (to appear in 2006).
- [15] A. McCallum and B. Wellner, ‘Toward conditional models of identity uncertainty with application to proper noun coreference’, in *IJCAI Workshop on Information Integration on the Web*, 2003, (2003).
- [16] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, ‘Finding predominant word senses in untagged text’, in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL’04)*, pp. 279–286, (2004).
- [17] P. Mika, ‘Bootstrapping the foaf-web: an experiment in social networking network mining’, in *Proceedings of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, (2004).
- [18] Ted Pedersen, Amruta Purandare, and Anagha Kulkarni, ‘Name discrimination by clustering similar contexts’, in *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, (2005).
- [19] V.V. Raghavan and S.K.M. Wong, ‘A critical analysis of vector space model for information retrieval’, *Journal of the American Society for Information Science*, **37**(5), 279–287, (1986).
- [20] Mehran Sahami and Tim Heilman, ‘A web-based kernel function for matching short text snippets’, in *International Workshop located at the 22nd International Conference on Machine Learning (ICML 2005)*, (2005).
- [21] Hinrich Schütze, ‘Automatic word sense discrimination’, *Computational Linguistics*, **24**(1), 97–123, (1998).

History-Based Inside-Outside Algorithm

Heshaam Feili and Gholamreza Ghassem-Sani¹

Abstract. Grammar induction is one of the most important research areas of the natural language processing. The lack of a large Treebank, which is required in *supervised* grammar induction, in some natural languages such as Persian encouraged us to focus on *unsupervised* methods. We have found the Inside-Outside algorithm, introduced by Lari and Young, as a suitable platform to work on, and augmented IO with a history notion. The result is an improved unsupervised grammar induction method called History-based IO (HIO). Applying HIO to two very divergent natural languages (i.e., English and Persian) indicates that inducing more conditioned grammars improves the quality of the resultant grammar. Besides, our experiments on ATIS and WSJ show that HIO outperforms most current unsupervised grammar induction methods.

1 INTRODUCTION

With the recent increasing interest in statistical approaches to natural language processing, *corpus* based linguistics has become a hot topic. This is due to the fact that computer based texts are available more than ever before, and easier to use for various data tasks. The success of part-of-speech tagging by using the Hidden Markov Model (HMM) [1, 2] also attracted the attention of computational linguists to the lexical analysis, language modeling, and machine translation by using various statistical methods [3, 4].

Manual design and refinement of a natural language grammar is a difficult and time-consuming task, and requires a large amount of skilled efforts. A hand-crafted grammar is not usually completely satisfactory, and frequently fails to cover many unseen sentences. Automatic acquisition of grammars is a solution to this problem. With the increasing availability of large, machine-readable, parsed corpora such as Penn Treebank [5], there have been numerous attempts to automatically derive a context free grammar by using such corpora [6].

Based on the level of supervision, which is used by different algorithms, grammar induction methods are divided in three main categories: *supervised*, *semi-supervised*, and *unsupervised*. Here, we present a novel unsupervised algorithm named **History-Based Inside-Outside** (HIO) as an extension of the well-known unsupervised **Inside-Outside** algorithm [7]. The experimental results of applying HIO to both English and Persian languages are also demonstrated and compared with that of several other unsupervised approaches.

2 PREVIOUS WORKS

In unsupervised grammar induction methods only tagged sentences without any bracketing information or other supervised information are used. Based on the Expectation Maximization (EM) algorithm, Lari and Young, proposed what they called the *Inside-outside* (IO) algorithm, that constructs a grammar from an unbracketed corpus [7]. The algorithm will converge to a local optimum when used to iteratively re-estimate probabilities on a training corpus in a manner, which maximizes the likelihood of the training corpus, given the grammar. This method is so far one of the basic algorithms for unsupervised automatic learning of grammars [8]. Also, Stolcke and Omohundro induced a small and artificial context free grammar with chunk-merge systems [9]. The results of these approaches for completely unsupervised acquisition showed that they are generally ineffective.

There are also other works to improve the quality of the unsupervised induction methods by considering some limitation or additional information. Magerman and Weir use a distituent grammar to eliminate undesirable rules [10]. Carroll and Charniak, restrict the set of non-terminals that may appear on the right hand side of rules with a given left hand side [11].

One of the most promising classes of unsupervised induction algorithms is based on particular distribution of words in sentences, and uses some distributional evidences to identify constituent structure [12]. Here, the main idea is that sequences of words (or tags) generated by the same non-terminal normally appear in similar contexts [13, 14]. Klein and Manning have introduced a new distributional method for inducing a bracketed tree structure of the sentence, with a dependency model to induces a word-to-word dependency structure [12, 14]. By combining these two models, they have achieved the best result in unsupervised grammar induction so far. Other dependency models with weaker results were presented by [11, 15, 16].

Alignment Based Learning (ABL) is a learning paradigm that can be regarded as a distribution based method. It is based on the principle of substitutability, whereby two constituents are of the same type, and then they could be substituted [17, 18]. Also, Adriaans presents EMILE, which initially used some aspects of supervision, but in later work is modified to be completely unsupervised [19]. Both the ABL and EMILE techniques look for *minimal pairs*; a specific form of distributional learning, where the contexts are the rest of the sentence.

Although supervised methods outperform current unsupervised induction algorithms with a relatively large gap, there are still compelling motivations for working on unsupervised methods [20]. Building supervised training data requires considerable resources, including time and linguistic expertise. This problem is more complicated when we deal with languages other than English,

¹ Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, emails: {hfaili@mehr.sharif.edu, sani@sharif.edu}

which usually lack the necessary resources. Furthermore, the resulting hand-crafted treebank may be too susceptible to restriction to a particular domain, application, or genre [21].

On the other hand, applying probabilistic model to natural languages has been investigated in several works where the independence of the input sentence and its context is assumed in parsing [1, 22]. In fact, most works have used even stronger independence assumptions. For instance, the PCFG model assumes the independence of the probability of each constituent and its neighboring constituents [1]. On the other hand, there are some richer models of context that incorporate some additional information with the probability of each constituent and present a way of calculating the probability model more accurately [22, 23].

There have been some promising works adopted the history based grammar induction methods. For instance, *Pearl* is a probabilistic parser that is more sensitive to the model of context [24]. Using supervised learning methods; *Pearl* acquired 88% of bracketing accuracy.

Another important work, which increases the dependencies on the context, is the *history-based parser* that was originally developed by the researchers at IBM [23, 25]. In these models, the parse-tree representation was enriched in a couple of ways: non-terminal labels were augmented by some extra information such as lexical items and head word. An improvement from 59.8% to 74.6% in parsing accuracy was reported by using this model [23].

The idea of adding the parent of each non-terminal as the conditioning information to the grammar rules was also mentioned in [22, 26, 27]. Replacing $P(\alpha \rightarrow \beta | \alpha)$ by $P(\alpha \rightarrow \beta | \alpha, \text{Parent}(\alpha))$, where $\text{Parent}(\alpha)$ is the non-terminal dominating α , leads to an improvement from 69.6% / 73.5% to 79.3% / 80.1% of the precision/recall metrics. In this paper, we introduce an extension of the IO algorithm augmented by the history notion, and apply it to two very different languages (i.e., English and Persian).

3 HIO ALGORITHM

In this section, we explain HIO, a new estimation model to induce an extended form of a PCFG by using the traditional IO algorithm. Here, the output of IO algorithm and other partial information such as inner and outer probabilities are used as the input data. This information is necessary to calculate the history-based model. The main idea of HIO is to decompose the output probabilities of rules extracted from IO, over their parent non-terminal. This decomposition can guide the parser toward better decisions in analyzing its input sentences.

In order to incorporate the parent non-terminal into the Chomsky normal form, we use the following notation to state the probability of using rule $i \rightarrow jk^2$:

$$A[i, j, k] = P(i \rightarrow jk | i \text{ used in derivation}, C = \text{Parent}(i), C \text{ used in derivation}) \quad (1)$$

Considering the conditional probability:

$$\begin{aligned} p(i \rightarrow jk | i \text{-used}, &= \text{parent}(i), -\text{used}) = \\ p(i \rightarrow jk, C = \text{Parent}(i) | i \text{-used}, C \text{-used}) \\ p(C = \text{parent}(i)) \end{aligned} \quad (2)$$

and $C = \text{Parent}(i)$, we can infer:

$$\forall U (C \rightarrow U i \text{ or } C \rightarrow i U) \quad (3)$$

² For the sake of simplicity, we ignore the assumption of using grammar G in all probabilities. Also, in the remainder of the paper, we use “i-used” to show that non-terminal i has been used in the derivation process.

$$\begin{aligned} p(C = \text{Parent}(i)) &= P(\forall U, C \rightarrow U i) + P(\forall U, C \rightarrow i U) - \\ P(C \rightarrow i) \end{aligned} \quad (4)$$

In the original IO algorithm [7], matrix “ a ” is defined as follows:

$$a[i, j, k] = p(i \rightarrow jk | i \text{-used}) \quad (5)$$

From (3) and (4), we can derive:

$$\begin{aligned} \forall C, \forall i, & p(C = \text{parent}(i)) = \\ \sum_u a[c, u, i] + \sum_u a[c, i, u] - a[c, i, i] \end{aligned} \quad (6)$$

By the assumption of having observation O , we obtain:

$$p(C \Rightarrow^* O(s) \dots O(t) | S \Rightarrow^* O, G) = e(s, t, C) \cdot f(s, t, C) / P \quad (7)$$

where $P = p(S \Rightarrow^* O | G)$, and $e(s, t, C)$ and $f(s, t, C)$ are the inner and outer probabilities, respectively.

But:

$$e(s, t, i) = \sum_{j, k} [\sum_{r=s}^{t-1} a[i, j, k] \cdot e[s, r, j] \cdot e(r+1, t, k)] \quad (8)$$

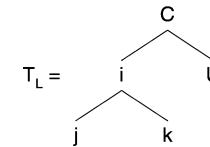
Thus, if at the first step of derivation starting from non-terminal C , rule “ $C \rightarrow i U$ ” is used, and by considering definitions of the inner probability (i.e., eq. 8) and matrix a (i.e., eq. 5), we can derive:

$$\begin{aligned} p(C \rightarrow iU \Rightarrow O(s) \dots O(t) | S \Rightarrow O, G) &= \\ \frac{1}{P} \sum_{r=s}^{t-1} \sum_{v=r}^{t-1} a[C, i, U] \cdot e(s, r, i) \cdot e(r+1, t, U) \cdot f(s, t, C) \quad \forall i, U, t > s \end{aligned} \quad (9)$$

Supposing that in the next step of the derivation, rule “ $i \rightarrow jk$ ” is used, the following equation holds:

$$\begin{aligned} p(C \rightarrow iU \rightarrow (jk)U \Rightarrow O(s) \dots O(t) | S \Rightarrow O, G) &= \\ \frac{1}{P} \sum_{r=s}^{t-1} \sum_{v=r}^{t-1} \sum_{w=v+1}^{t-1} a[C, i, U] \cdot e(s, r, i) \cdot e(r+1, t, U) \cdot f(s, t, C) \quad \forall i, j, U, t > s \end{aligned} \quad (10)$$

By naming the next left branching tree as T_L , we can rewrite equation (10) as follows:



$$p(T_L, i\text{-used}, C\text{-used}) = \text{equation (10)} \quad (11)$$

Therefore:

$$p(T_L | i\text{-used}, C\text{-used}) = \text{equation (10)} / P(i\text{-used}, C\text{-used}) \quad (12)$$

By assuming the independence of using non-terminals i and C , we have:

$$p(i\text{-used}, C\text{-used}) = p(i\text{-used}) \cdot p(C\text{-used}) \quad (13)$$

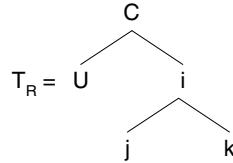
And from (12) and (13), we get the following equation:

$$p(T_L | i\text{-used}, C\text{-used}) = \text{equation(10)} / (p(i\text{-used}) \cdot p(C\text{-used})) \quad (14)$$

Equation (15) can be derived in a way similar to that of (10):

$$\begin{aligned} p(C \rightarrow U_i \rightarrow U(jk) \Rightarrow O(s) \dots O(t) | S \Rightarrow O, G) &= \\ \frac{1}{P} \sum_{r=s}^{t-1} a[C, U, i] \cdot \sum_{v=r+1}^{t-1} a[i, j, k] \cdot e(r, v, j) \cdot e(v+1, t, k) \cdot e(s, r, U) \cdot f(s, t, C) \quad \forall i, j, U, t > s \end{aligned} \quad (15)$$

And by defining T_R , as follows:



$$p(T_R \mid i\text{-used}, C\text{-used}) = \text{equation(15)} / (P(i\text{-used}) \cdot P(C\text{-used})) \quad (16)$$

Now, we can evaluate the main equation:

$$\begin{aligned} p(i \rightarrow jk, C = \text{Parent}(i) \mid i\text{-used}, C\text{-used}) &= \\ \Sigma_U p(i \rightarrow jk, (C \rightarrow U_i \text{ or } C \rightarrow iU) \mid i\text{-used}, C\text{-used}) &= \\ \Sigma_U p(i \rightarrow jk, C \rightarrow U_i \mid i\text{-used}, C\text{-used}) + \\ P(i \rightarrow jk, C \rightarrow iU \mid i\text{-used}, C\text{-used}) - \\ p(i \rightarrow jk, C \rightarrow ii \mid i\text{-used}, C\text{-used}) \end{aligned} \quad (17)$$

T_L and T_R are equal to:

$$T_L = \{ C \rightarrow i \text{ } U, i \rightarrow jk \}, T_R = \{ C \rightarrow U \text{ } i, i \rightarrow jk \} \quad (18)$$

By replacing T_L and T_R in (17), one gets:

$$\begin{aligned} p(i \rightarrow jk, C = \text{Parent}(i) \mid i\text{-used}, C\text{-used}) &= \\ \text{eq.(14)} + \text{eq.(16)} - \\ p(C \rightarrow i \text{ } i \rightarrow (jk) \text{ } i \rightarrow (jk)(jk) \mid i\text{-used}, C\text{-used}) \end{aligned} \quad (19)$$

The last term of (19), can be computed by using (10), where U is initialized to i and later expanded by $i \rightarrow jk$. Thus:

$$\begin{aligned} p(C \rightarrow i \text{ } i \rightarrow (jk) \text{ } i \rightarrow (jk)(jk) \Rightarrow O(s) \dots O(t) \mid S \Rightarrow O, G) &= \\ \frac{1}{P} \sum_{r=s}^{t-1} a[C, i, i].f(s, t, C). & \\ \sum_{v=s}^{r-1} a[i, j, k].e(s, v, j).e(v+1, r, k). & \\ \sum_{w=r+1}^{t-1} a[i, j, k].e(r+1, w, j).e(w+1, t, k) \quad \forall i, j, t > s & \end{aligned} \quad (20)$$

The third line of equation (20) is equal to $e(s, r, j)$ and the fourth line is $e(r+1, t, i)$ (see [7]).

The probability of using any non-terminal i in a derivation, is computed as in [7]:

$$p(i\text{-used}) = \sum_{s=1}^T \sum_{t=1}^T \frac{1}{P} e(s, t, i).f(s, t, i) \quad (21)$$

Considering equations (2), (5), (9), (20) and (21), we can compute the probability mentioned in (1), which is the main parameter of the history-based model:

$$\begin{aligned} p(i \rightarrow jk \mid i\text{-used}, C = \text{Parent}(i), C\text{-used}) &= \\ (\text{eq.(14)} + \text{eq.(16)} - \text{eq.(20)}) / \text{eq.(21)} & \end{aligned} \quad (22)$$

The last equation, denoted by $A/C, i, j, k$, is the main estimation formula used by HIO for supporting the history-based model.

In a similar manner, the second form of Chomsky rules ($i \rightarrow m$), can be extended to support probability of using such rules, considering parent non-terminals as well.

After evaluation of parameter matrices a and b in the traditional IO algorithm, we evaluate matrices A and B . The main iteration of inside-outside algorithm will be terminated when changes in the overall probability of observations is less than a pre-defined threshold.

The HIO model induced by the mentioned approach assumes parent non-terminals in the parsing. Therefore, the algorithm which is used during the parsing time, should consider the parent non-

terminal too. An extension of PCYK algorithm described in [27] is used for this purpose.

4 EXPERIMENTAL RESULTS

Two kinds of experiments are presented in this section. At first the results of evaluating HIO on a number of English data sets are demonstrated. Then the results of applying HIO to Persian, which is essentially very different from English, are also discussed. HIO was tested on both ATIS [28] and Wall Street Journal [6]. In order to be able to compare with others' work, we selected these data sets for testing the induced grammar. We used only POS tag sequences as the lexical information of the training and testing data sets.

We executed two different experiments on English sentences. At first, as in other works, ATIS was divided into two distinct sets: the *training* set with approximately 90% of data and the *test* set (i.e., remaining 10%). Note that although our approach is unsupervised and does not need a bracketing data set, we need the tree style of syntactic information of the test data set for the evaluation purpose. The results were computed using the so-called ten fold cross validation method. In the second experiment, 7400 sentences shorter than 11 words of the Wall Street Journal (WSJ-10) were selected. This data set was the main corpus used in training of HIO, and the induced grammar was tested on both ATIS and WSJ, too.

The initial grammar of IO is a full grammar, which contains all possible CNF rules, with random probabilities assigned to each rule. To alleviate the algorithm from any possible bias, the process was repeated one hundred times with different initial probabilities.

The outputs of all experiments were evaluated on a test corpus by using the extended PCYK parsing algorithm. This algorithm gets the extended model of PCFG and a test sentence, generates all possible parses of the input sentence in a dynamic programming manner, and selects the most probable parse. Then the parsed sentences were evaluated by comparison against the corresponding treebank of the same corpus.

In the first experiment, we selected spoken-language transcription of the Texas Instruments subset of the Air Travel Information System (ATIS) corpus [28]. This corpus, which has been automatically labeled, analyzed and manually checked, is included in Penn Treebank II [5]. There are two different pieces of labeling information in the Treebank: a part of speech tag and a syntactic labeling. We used 577 sentences of the corpus with 4609 words. We ran the experiments with both 5 and 15 non-terminals³, and every experiment was also repeated one hundred times. The means of the results from both IO and HIO, and the standard deviations of the latter are shown in table 1.

Table 1. the results of IO and HIO on ATIS data set

No. of non-terminals = 5			
	UP	UR	F1
IO	30.03	25.27	27.45
HIO	42.54(3.15)	38.95(5.8)	40.67(6.8)
No. of non-terminals = 15			
	UP	UR	F1
IO	42.19	35.51	38.56
HIO	46.85(3.2)	40.9(5.9)	43.67(3.9)

As it is shown, HIO significantly outperforms IO on the mentioned corpus. Moreover, the ratio of improvement of IO is

³ By considering the NULL non-terminal, the numbers of used non-terminal in these experiments are 6 and 16 respectively.

very sensitive to the number of non-terminals. In other words, the number of non-terminals used in the IO algorithm has a remarkable impact on the accuracy of the induced grammar; while HIO's accuracy improvement is much less dependent on this factor. This is due to the structure of the richer model used by HIO⁴.

To compare HIO with other unsupervised methods, we collected the results of testing several different approaches on ATIS data set in table 2: EMILE [19], ABL [17], CDC with 40 iterations [13], and CCM [14]. LEFT and RIGHT are left- and right-branching baselines applied to ATIS. The results of LEFT and RIGHT baselines have been taken from [14].

HIO was trained on two different data sets. It was trained on WSJ-10, and tested on ATIS data set. Then it was also trained on 90% of ATIS, and tested on the remaining 10%. The second experiment was evaluated by the ten fold cross validation method. HIO-WSJ and HIO-ATIS respectively correspond to these two experiments.

Table 2. the results of different approaches on ATIS data set

Method	UP	UR	F1
EMILE	51.59 (2.70)	16.81 (0.69)	25.35 (1.00)
ABL	43.64 (0.02)	35.56 (0.02)	39.19 (0.02)
CDC-40	53.4	34.6	42.0
CCM	55.4	47.6	51.2
LEFT	19.89	16.74	18.18
RIGHT	39.9	46.4	42.9
IO	42.19	35.51	38.56
HIO-WSJ	45.2 (2.6)	41.6 (1.5)	43.32 (1.9)
HIO-ATIS	46.85 (3.2)	40.9 (5.9)	43.67 (4.15)

As it's shown, HIO's output is superior to that of all other mentioned works except CCM, which seems to be the current state of the art in unsupervised grammar induction. Also, results of HIO-ATIS are better than that of HIO-WSJ because the test and training data set have been selected from the same corpus.

We also tested HIO on WSJ in order to compare it with other published works, especially with CCM [14]. The mean value of F1 score for HIO and other different methods on WSJ-10 are shown in Figure 1⁵. RANDOM selects a tree uniformly at random from the set of binary trees. DEP-PCFG is the result of duplicating the experiments of [11], using EM to train a dependency structured PCFG. SUP-PCFG is a supervised PCFG parser trained on a 90-10 split of WSJ-10 data set, using the treebank grammar with the Viterbi parse right-binarianized. UBOUND is the upper bound of how well a binary system can cope with the treebank sentences that are generally flatter rather than being binary, limiting the maximum achievable precision. As it is shown, although HIO outperforms the right-branching baseline, CCM still has the best rank among unsupervised methods.

We have also tested HIO on Persian, which is linguistically very divergent from English [3]. In order to test HIO on Persian, we manually produced a treebank adopting different notions of the Penn Treebank. We chose a data set called *Peykareh* as the initial corpus for our experiments [29]. This corpus has more than 7000 sentences collected from formal newsletters in Persian. The tag set that was used in the corpus is the same as one used by [30, 31]; however some of the tags were merged in order to be used in the

⁴ Note that HIO uses an extended PCFG model in which each rule is associated with the parent of its left-hand side.

⁵ The higher results on WSJ, compared with that of ATIS, are due to the tendency of ATIS to have shorter constituents, especially with one word. HIO and IO do not bracket single words, and that is why the F1 measure (especially recall) decreases.

syntactical analysis⁶, remaining only 18 POS tags. We selected 2200 sentences shorter than 11 words from *Peykareh* for both HIO and IO. Table 3, shows the results of these experiments.

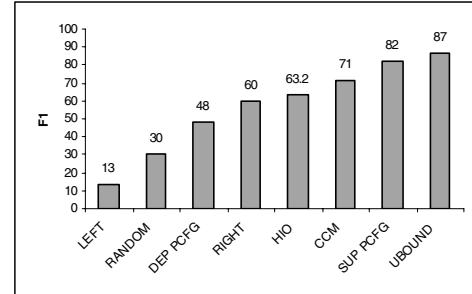


Figure 1. F1 score for various models on WSJ-10

Table 3. the results of IO and HIO on Persian corpus with sentences shorter than 11 words

No. of non-terminals = 5		
	UP	UR
IO	33.25(0.64)	31.93(0.87)
HIO	41.26(1.16)	38.59(2.3)
No. of non-terminals = 15		
	UP	UR
IO	44.35(0.5)	40.1(0.68)
HIO	52.5(1.29)	50.28(1.7)

As it is shown, in Persian too, HIO outperforms IO by more than 20% with respect to the F1 measure. Figure 2 compares the effect of the grammar size (i.e., the number of its non-terminals) on the quality of the induced grammar (again based on F1) in Persian with that of English. This comparison implies that the grammar size affects Persian more than English. In other words, a larger grammar is required to model Persian. That is mainly because Persian is a free-word-order language, and thus harder to model. Therefore, HIO can achieve even better results on free-word order languages by using larger grammars.

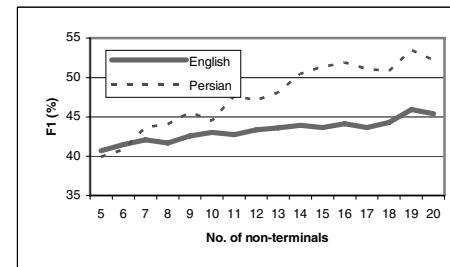


Figure 2. The effect of the grammar size on F1 measure in English and Persian

5 CONCLUSION

We showed that adding the parent non-terminal label to the rule assumptions increases the quality of output grammar. In other words, by relaxing the flawed independence assumption, we can construct a richer model of unsupervised grammar induction method. It might be the case that augmenting the model with some

⁶ The tag set presented by [29, 31] was used only for morphological analysis tasks; the set needed to be re-arranged and merged in a syntactical analysis view.

other conditions further improves the accuracy of the results. However, adding extra conditions to the model may also cause an exponential increase in the size of the output grammar, which hampers the parsing process.

We introduced HIO, a new approach based on the well-known IO algorithm for inferring stochastic context-free grammars. We relaxed the independence assumptions often used with the PCFG rules in the parsing process by adding some extra information about the context to the derivation process. Here, the parent non-terminal, which dominates the PCFG rule, was chosen as the contextual information.

The new method was applied to both English and Persian languages. The results showed a significant superiority over almost all other unsupervised methods. The results on Persian was even more prominent though it is a free-word-order language and harder to model. CCM, a distributional algorithm based on the idea that similar words occur in similar contexts [14], is the only unsupervised method that outperforms HIO on English. However, as it was also pointed out by Clark [11], we think that distributional methods might not be applicable to free-word-order languages, and intend to verify that.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers of the paper for their invaluable suggestions and comments. This work has been partially supported by the Iranian Telecommunication Research Center (ITRC).

REFERENCES

- [1] Charniak, E., "Statistical techniques for natural language parsing", AI Magazine, Vo. 18, No. 4, pp. 33-44, Winter 1997.
- [2] Church, K., "A stochastic parts program and noun phrase parser for unrestricted text", In the Proceedings of the Second Conference on Applied Natural Language Processing, pp. 136-143, 1988.
- [3] Feili, H. and Ghassem-Sani, G., "An Application of Lexicalized Grammars in English-Persian Translation", Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Universidad Politecnica de Valencia, Spain, pp. 596-600, 2004.
- [4] Charniak, E., "Statistical Language Learning", Cambridge, London, UK, MIT Press, 1996.
- [5] Mitchell P., Marcus, B., Santorini, ce., and Marcinkiewicz, M. A., "Building a large annotated corpus of English: the Penn Treebank", Computational Linguistics, Vol. 19, pp. 313-330, 1993.
- [6] Pereira, F. and Y. Schabes, "Inside-Outside reestimation from partially bracketed corpora", In proceeding of 30th annual Meeting of the ACL, pp. 128-135, 1992.
- [7] Lari, K. and Young, S.J., "The estimation of stochastic context-free grammar using the inside-outside algorithm", Computer Speech and Language, Vol. 4, pp. 35-56, 1990.
- [8] Briscoe, T., and Waegner, N., "Robust stochastic parsing using the inside-outside algorithm". In AAAI-92 Workshop on Statistically Based NLP Techniques, 1992.
- [9] Stolcke, A., and Omohundro, S. M., "Inducing probabilistic grammars by Bayesian model merging". In Grammatical Inference and Applications: Proceedings of the Second International Colloquium on Grammatical Inference. Springer Verlag, 1994.
- [10] Magerman, D. M. and Marcus, M. P., "Parsing a natural language using mutual information statistics", In Proceedings of the Eighth National conference on Artificial Intelligence, August, 1990.
- [11] Carroll, G. and Charniak, E., "Two experiments on learning probabilistic dependency grammars from corpora". In C. Weir, S. Abney, R. Grishman, and R. Weischedel, editors, Working Notes of the Workshop Statistically Based NLP Techniques, pages 1-13. AAAI Press, 1992.
- [12] Klein, D., and Manning, C. D., "Natural language grammar induction using a constituent-context model", In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14 (NIPS 2001), Vol. 1, 35-42, MIT Press, 2001.
- [13] Clark, A., "Unsupervised Language Acquisition: Theory and Practice", PhD thesis, University of Sussex, 2001.
- [14] Klein, D., "The Unsupervised Learning Of Natural Language Structure", PhD Thesis, Department of Computer Science, Stanford University, 2005.
- [15] Yuret, D., "Discovery of Linguistic Relations Using Lexical Attraction", PhD thesis, MIT, 1998.
- [16] Paskin, M. A., "Grammatical bigrams", In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14, Cambridge, MA. MIT Press, 2002.
- [17] Van Zaanen, M., "ABL: Alignment-Based Learning", In COLING 2000, pp. 961-967, 2000.
- [18] Van Zaanen, M. and Adriaans, P. W., "Comparing Two unsupervised Grammar Induction Systems: Alignment-Based Learning vs. EMILE", Technical Report: TR2001.05, School of Computing, University of Leeds, 2001.
- [19] Adriaans, P., and Haas, E., "Grammar induction as sub-structural inductive logic programming". In J. Cussens (Ed.), Proceedings of the 1stWorkshop on Learning Language in Logic, 117-127, Bled, Slovenia, 1999.
- [20] Marcken, C., "Unsupervised Language Acquisition", PhD. thesis, Department of Electrical Engineering and Computer Science, MIT, 1996.
- [21] Kehler, A. and Stolcke, A., Preface, In A. Kehler and A. Stolcke, editors, "Unsupervised Learning in Natural Language Processing", Association for Computational Linguistics, Proceedings of the workshop, 1999.
- [22] Johnson, M., "The Effect of Alternative Tree Representations on Tree Bank Grammars", In D.M.W. Powers (ed.) NeMLA/P3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning, ACL, pp. 39-48, 1998.
- [23] Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R. and Roukos, S., "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing", In the Proceedings of the 5th DARPA Speech and Natural languages Workshop, Harriman, NY, 1992.
- [24] Magerman, D. and Marcus, M., "Pearl: A Probabilistic Chart Parser", In the Proceedings of the 1991 European ACL conference, Berlin, Germany, 1991.
- [25] Jelinek, F., Laferty, J. D., Magerman, D., Mercer, R., Ratnaparakhi, A. and Roukos, S., "Decision-Tree Parsing using Hidden Derivation Model", In the Proceedings of the 1994 Human Language Technology Workshop, pp. 272-277, 1994.
- [26] Feili, H. and Ghassem-Sani, G., "One Step toward a richer model of unsupervised grammar induction", Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2005), 21-23 September, 2005, Borovets, Bulgaria, pp. 197-203.
- [27] Feili, H. and Ghassem-Sani, G., "Unsupervised Grammar Induction Using History Based Approach", to appear in Computer Speech and Language Journal, Elsevier publishing, 2006.
- [28] Hemphill, C.T., Godfrey, J., and Doddington, G., "The ATIS spoken language systems pilot corpus", In DARPA Speech and Natural language Workshop, Hidden Valey, Pennsylvania, June 1990.
- [29] Bijankhan, M., "Naghshe Peykarehaye Zabani dar Neveshtane Dasture Zaban:Mo'arrefiye yek Narmafzare Rayane'i [the Role of Corpus in generating grammar: Presenting a computational software and Corpus]", Iranian Linguistic Journal, No. 19, Vol. 2, pp. 48-67, 2005 (in Persian).
- [30] Amtrup, Jan W., Rad, H. R., Megerdoomian, K., and Zajac, R., "Persian-English Machine Translation: An Overview of the Shiraz Project", NMSU, CRL, Memoranda in Computer and Cognitive Science (MCCS-00-319), 2000.
- [31] Megerdoomian, Karine, "Persian Computational Morphology: A Unification-Based Approach". NMSU, CRL, Memoranda in Computer and Cognitive Science (MCCS-00-320), 2000.

Shallow Semantic Parsing Based on FrameNet, VerbNet and PropBank

Ana-Maria Giuglea¹ and Alessandro Moschitti²

Abstract. This article describes a semantic parser based on FrameNet semantic roles that uses a broad knowledge base created by interconnecting three major resources: FrameNet, VerbNet and PropBank. We link the above resources through a mapping between Intersective Levin classes, which are part of PropBank's annotation, and the FrameNet frames. By using Levin classes, we successfully detect FrameNet semantic roles without relying on the frame information. At the same time, the combined usage of the above resources increases the verb coverage and confers more robustness to our parser. The experiments with Support Vector Machines on automatic Levin class detection suggest that (a) tree kernels are well suited for the task and (b) Intersective Levin classes can be used to improve the accuracy of semantic parsing based on FrameNet roles.

1 INTRODUCTION

Knowing the semantic roles played by the entities that appear in a sentence is of major importance for understanding its underlying meaning. The inherent word ambiguity can lead to very different readings of the same sentence. One important step towards deciding the correct reading is to find the sense of the verb in the sentence.

The previous point was highlighted by the results, obtained with and without the frame information during the Senseval-3 competition on FrameNet [10] role labeling task [17]. When such information was not used by the systems, the performance drop was more than 10 percent points. This is quite intuitive as the semantics of many roles strongly depend on the focused frame. Thus, we cannot expect a good performance when this information is not available.

A solution to this problem is the automatic frame detection. Unfortunately, our preliminary experiments showed that given a FrameNet (FN) predicate-argument structure, the task of identifying the associated frame can be performed with very good results when the verb predicates have enough training examples, but becomes very challenging otherwise. The predicates not yet included in FN, e.g. belonging to new application domains, are especially problematic since there is no training data available. In such cases the frame classifier reaches very low accuracy (under 50%).

We have thus studied new means of capturing the semantic context, other than the frame, which can be easily annotated on FrameNet and are available on a larger scale (i.e. have a better coverage). A very good candidate seems to be the Intersective Levin class information [2] that can be found as well in other predicate resources like PropBank and VerbNet.

In this paper:

¹ University of Rome "Tor Vergata": agiuglea@gmail.com

² University of Rome "Tor Vergata": moschitti@info.uniroma2.it

1. we employ SVM Tree Kernels and structural features [19] for automatic token-based verb classification³ of the Intersective Levin classes (ILCs);
2. we use the classifiers trained in step 1 to automatically annotate FrameNet with Levin class information which is needed for the semantic role labeling (SRL) task; and
3. we test the effectiveness of the ILC information in the FrameNet SRL task.

On FrameNet, we obtain the gold Levin class annotation through a mapping between Intersective Levin classes and FrameNet frames [7]. This mapping employs also the PropBank corpus [11] and the VerbNet lexicon [12].

In the remainder of this paper Section 2 and 3 introduce the resources used by our study, namely VerbNet, PropBank and FrameNet, Section 4 summarizes previous work on Levin verb sense disambiguation, Section 5 focuses on Tree Kernels and the features used while Section 6 contains a description of the performed experiments. Finally, Section 7 presents our results and conclusions.

2 LEVIN AND PROPBANK

Levin clusters [15] are formed according to diathesis alternation criteria which are variations in the way verbal-arguments are grammatically expressed when a specific semantic phenomenon arises. For example, two different types of diathesis alternations are the following:

- (a) Middle Alternation

[Subject, Agent The butcher] cuts [Direct Object, Patient **the meat**].
[Subject, Patient **The meat**] cuts easily.
- (b) Causative/inchoative Alternation

[Subject, Agent Janet] broke [Direct Object, Patient **the cup**].
[Subject, Patient **The cup**] broke.

In both cases, what is alternating is the grammatical function that the Patient role takes when changing from the transitive use of the verb to the intransitive one. The semantic phenomenon accompanying these types of alternations is the change of focus from the entity performing the action to the theme of the event.

Levin documented 79 alternations which constitute the building blocks for the verb classes. Although alternations are chosen as the primary means for identifying the classes, additional properties related to subcategorization, morphology and extended meanings of

³ The best semantic class is determined for the verb token given the local context of the phrase rather than using the set of verb occurrences across a corpus or a document (i.e. type-based classification).

verbs are taken into account as well. Thus, from a syntactic point of view, the verbs in one Levin class have a regular behavior, different from the verbs pertaining to other classes. Also, the classes are semantically coherent and all verbs belonging to one class share the same participant roles.

This constraint of having the same semantic roles is further ensured inside the VerbNet lexicon which is constructed based on a more refined version of the Levin classification called Intersective Levin classes [2]. The lexicon provides a regular association between the syntactic and semantic properties of each of the described classes. It also provides information about the syntactic frames (alternations) in which the verbs participate together with the set of possible semantic roles.

One corpus associated with the VerbNet lexicon is PropBank. The annotation scheme of PropBank ensures that the verbs belonging to the same Levin class and exhibiting the same diathesis alternations share similarly-labeled arguments. Inside one Levin class to one argument corresponds one semantic role numbered sequentially from Arg0 to Arg5. Higher numbered argument labels are less consistent and assigned per verb basis.

Besides semantic roles, PropBank is annotated also with Intersective Levin class information and contains gold parse trees. These features made PropBank very suitable for testing the effectiveness of our Tree Kernel approach for ILC detection. We were able to measure the accuracy of our machine learning algorithm in the presence of gold predicate-argument structures, which gave us a performance upper bound.

During the second step (i.e. annotating FrameNet with ILC), in order to train ILC classifiers also on FrameNet we need gold Intersective Levin class annotations. To achieve that we employed a semi-automatic algorithm that mapped FrameNet frames to Levin classes, thus assigning gold ILC to FrameNet. More details about FrameNet and the mapping algorithm are presented in the next section.

3 LEVIN AND FRAMENET

One of the goals of the FrameNet project is to design a hierarchical linguistic ontology that can be used for automatic processing of semantic information. This hierarchy contains an extensive semantic analysis of verbs, nouns, adjectives and situations in which they are used, called frames. The basic assumption on which the frames are built is that each word evokes a particular situation with specific participants [5]. The situations depict the entities involved and the roles they play. The word that evokes a particular *frame* is called *target word* or predicate and can be an adjective, noun or verb. The participant entities are defined using semantic roles and they are called *frame elements*.

Predicates belonging to the same FrameNet frame were proven [6] to have a coherent syntactic behavior that is also different from predicates pertaining to other frames. This finding is consistent with the assumption on which Levin's verb classification is build. This insight determined us to study the relation between FrameNet frames and Levin classes.

The Levin classes were constructed based on regularities exhibited at grammatical level and the resulting clusters were shown to be semantically coherent. As opposed, the FrameNet frames were build on semantic bases, by putting together verbs, nouns and adjectives that evoke the same situations. Although different in conception, the FrameNet verb clusters and VerbNet verb clusters have common properties:

1. Different syntactic properties between distinct verb clusters (as proven by the experiments in [6])
2. Shared sets of possible semantic roles for all verbs pertaining to the same cluster.

Having these insights, we have assigned a correspondent VerbNet class not to each verb predicate but rather to each frame. In doing this we have applied the simplifying assumption that a frame has a unique corresponding Levin class. Thus, we have created a one-to-many mapping between the Levin classes and the frames.

The mapping algorithm consists of three steps: (a) we link the frames and Intersective Levin verb classes that have the largest number of verbs in common and we create a set of pairs $\langle FN \text{ frame}, VN \text{ class} \rangle$ (see Table 1); (b) we refine the pairs obtained in the previous step based on diathesis alternation criteria, i.e. the verbs pertaining to the FN frame have to undergo the same diathesis alternation that characterize the corresponding VN class and (c) we manually check the resulting mapping.

INPUT
$VN = \{C C \text{ is a VerbNet class}\}$
$VN \text{ Class } C = \{v v \text{ is a verb of } C\}$
$FN = \{F F \text{ is a FrameNet frame}\}$
$FN \text{ frame } F = \{v v \text{ is a verb of } F\}$
OUTPUT
$Pairs = \{(F, C) F \in FN, C \in VN : F \text{ maps to } C\}$
COMPUTE PAIRS:
<i>Let Pairs = \emptyset</i>
<i>for each $F \in FN$</i>
<i>(I) compute $C^* = \arg \max_{C \in VN} F \cap C$</i>
<i>(II) if $F \cap C^* \geq 3$ then $Pairs = Pairs \cup \langle F, C^* \rangle$</i>

Table 1. Linking FrameNet frames and VerbNet classes.

During the second step of the mapping we make use of the property (2) of the Levin classes and FN frames presented in this section. According to this property, all verbs pertaining to one frame or Levin class have the same participant roles. Thus, a first test of compatibility between a frame and a Levin class is that they share the same participant roles. As FN is annotated with frame-specific semantic roles, we manually mapped these roles into the VN set of thematic roles. Given a frame, we assigned thematic roles to all frame elements that are associated with verbal predicates. For example the *Speaker*, *Addressee*, *Message* and *Topic* roles from the *Telling* frame were respectively mapped into the *Agent*, *Recipient*, *Theme* and *Topic* theta roles.

After the role matching, the mapping algorithm checks both the syntactic and semantic consistency by comparing the role frequency distributions on different syntactic positions for the two candidates. More details are given in [7]. We mention that the algorithm identifies correctly the cases for which our simplifying assumption does not hold, having an overall accuracy of 89.6%.

Having gold ILC annotation on FrameNet allows us to train Intersective Levin class classifiers also on this corpus. In this way, we can extend the verb coverage to encompass both PropBank and FrameNet.

In the next sections we describe our approach on ILC automatic detection and also some of the literature work on this subject.

4 PREVIOUS WORK ON LEVIN CLASS DETECTION

Levin's verb classification is based on straightforward syntactic criteria which makes it especially appealing for automation. As a consequence, it was used in many different studies ranging from machine translation [3] and information retrieval [16] to automatic acquisition of lexical semantic information and creation of dictionaries [4].

Regarding automatic verb classification, most of the previous studies focused on type-based classification. We mention Merlo and Stevenson [18, 21] who use grammatical features to classify verbs into three classes: unergative, unaccusative and object-drop. These classes comprise several Levin classes and were chosen because they participate in similar alternations (i.e. transitive alternations) but they assign different thematic roles to their arguments.

In their study, Merlo and Stevenson investigate to what extent features extracted from the predicate argument structures are useful for disambiguating among unergative, unaccusative and object-drop. Some of the most successful features used were Causativity and Transitivity. Causativity feature is a marker for verbs participating in causative alternations (e.g. the sentences of the example (b) of Section 2) and measures how many times the same noun occurs as subject and as object of the verb (i.e. the degree of overlap). Transitivity is a binary feature that marks whether the verb is used in a transitive or intransitive form. We will show in the next section that structural features capture both the causative and transitive markers.

Other studies use subcategorization information and selectional restrictions to cluster verbs into Levin compatible classes [8]. The resulting clusters are measured against Levin's classification having a 61% match. Although it is counterintuitive, the selectional preferences for the arguments in the subcategorization frames seem to have a negative impact on performance.

The subcategorization feature is used also in Lapata and Brew's studies [14, 13] which focus on dative and benefactive alternations. They view the choice for a class as being estimated by the joint probability $P(\text{verb}, \text{syntactic frame}, \text{class})$ and they design a distributional model that uses the BNC corpus. The results obtained are very promising considering that only subcategorization information is used (74.6% accuracy for genuinely ambiguous verbs). Also, Lapata and Brew incorporate their distributional model as prior knowledge in a naive Bayes classifier for performing token-level disambiguation. Unfortunately, their results are reported per syntactic frame which makes them hard to compare with ours.

Overall, the previous studies are restricted in scope by focusing only on specific aspects or alternations involved in the Levin's verb classification. One novelty of our approach is the fact that we use corpora that are annotated with Levin class information. Thus, we are able to appreciate better to what extent our method is applicable for Levin class disambiguation in general. Our analysis is conducted on a number of 179 classes that had training examples in the corpora.

In the following section we will present the algorithm and the features used by our model. We will show that structural features include some of the best features developed in the literature like causativity, transitivity and also, the very important subcategorization information.

5 Tree Kernels and Feature Space

The main idea of tree kernels is the modeling of a $K_T(T_1, T_2)$ function which computes the number of common substructures between two trees T_1 and T_2 . Thus, we can use, as features, structures drawn

directly from the syntactic parse tree of the sentence.

The kernel that we employed in our experiments was devised in [19]. We used a reduced version of the predicate-argument structure (Figure 1) which contains also the headwords of the arguments, useful for representing the selectional preferences. This feature is a variant of the SCF feature from [19], but in a reduced format (hereafter called SCF-reduced). In the following figure we present an example of the SCF-reduced feature constructed for the sentences of the example (b) of Section 2.

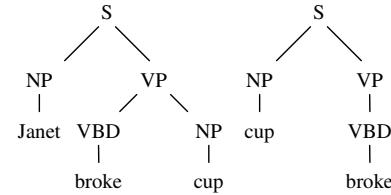


Figure 1. SCF-reduced of the sentences from the example (b) of Section 2

The trees in Figure 1 have respectively 17 and 10 substructures from which 3 are shared (Figure 2).

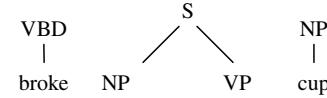


Figure 2. Common substructures of the sentences of Figure 1.

We note that the overlap between "cup" used on the subject position and "cup" used on the object position increases the similarity measure between the two sentences of the example (b). Thus, the Causativity feature from [18] is subsumed by the SCF-reduced feature. Other substructures contain the subcategorization frame with or without the verb marked (Figure 3). As in general the subcategorization frame embeds also the transitivity or intransitivity marker, we conclude that both the Transitivity feature [18] and Subcategorization feature [14, 8] are subsumed by SCF-reduced.

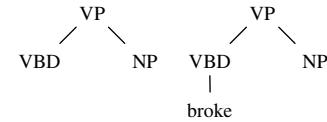


Figure 3. Substructures embedding the Subcategorization feature.

The next section shows that SCF-reduced detects ILCs with high accuracy and is very robust with respect to automatic parse trees and different corpora.

6 EXPERIMENTS

The aim of this research is to show that the ILC feature is very effective for the FrameNet semantic role labeling (SRL) task. For this purpose we carried out several tests.

During our first experiment set we trained (1) an ILC multiclassifier from FN, (2) an ILC multiclassifier from PB and (3) a frame multiclassifier from FN. We compared the results obtained when trying to classify ILCs with the results obtained when classifying frame. We show that ILCs are easier to detect than FN frames.

	run- 51.3.2	cooking -45.3	characterize- 29.2	other_cos- 45.4	say- 37.7	correspond- 36.1	Multiclassifier
PB #Train Instances	262	6	2,945	2,207	9,707	259	52,172
PB #Test Instances	5	5	134	149	608	20	2,742
PB Results	75	33.33	96.3	97.24	100	88.89	92.96
FN #Train Instances	5,381	138	765	721	1,860	557	46,734
FN #Test Instances	1,343	35	40	184	1,343	111	11,650
FN Results	96.36	72.73	95.73	92.43	94.43	78.23	92.63

Table 2. Argument classifier F1s and the overall multiclassifier accuracy for ILC labeling.

	Body_part	Crime	Degree	Agent	Multiclassifier
FN #Train Instances	1,511	39	765	6,441	102,724
FN #Test Instances	356	5	187	1,643	25,615
LF+Gold Frame	90.91	88.89	70.51	93.87	90.8
LF+Gold ILC	90.80	88.89	71.52	92.01	88.23
LF+Automatic Frame	84.87	88.89	70.10	87.73	85.64
LF+Automatic ILC	85.08	88.89	69.62	87.74	84.45
LF	79.76	75.00	64.17	80.82	80.99

Table 3. Argument classifier F1s and the overall multiclassifier accuracy for FN semantic role labeling.

Our second set of experiments regards the automatic labeling of FN semantic roles on FN corpus when using as features: gold frame, gold ILC, automatically detected frame and automatically detected ILC. We show that in all situations in which the ILC feature is used, the accuracy loss, compared to the usage of the frame feature, is negligible. We thus show that the ILC can successfully replace the frame feature for the task of semantic role labeling.

Another set of experiments regards the generalization property of the ILC. We show the impact of this feature on SRL when very few training data is available and its evolution when adding more and more training examples. We again perform the experiments for: gold frame, gold ILC, automatically detected frame and automatically detected ILC.

Finally, we simulate the difficulty of free text by annotating PB with FN semantic roles. We used PB because it is different from FN from a domain point of view. This characteristic makes PB a complex test bed for semantic role models trained on FN. In the following section we present the results obtained for each of the experiments mentioned above.

6.1 Experimental setup

The corpora available for the experiments were PB and FN. PB contains about 54,900 sentences and gold parse trees. We used sections from 02 to 22 (52,172 sentences) to train the ILC classifiers and section 23 (2,742 sentences) for testing purposes.

For the experiments on FN corpus, we extracted 58,384 sentences from the 319 frames that contain at least one verb annotation. There are 128,339 argument instances of 454 semantic roles. Only verbs are selected to be predicates in our evaluations. Moreover, as there is no fixed split between training and testing, we randomly selected 20% of sentences for testing and 80% for training. The sentences were processed using Charniak's parser [1] to generate parse trees automatically.

The classification models were implemented by means of the SVM-light-TK software available at <http://ai-nlp.info.uniroma2.it/moschitti> which encodes tree kernels in the SVM-light software [9]. We used the

default parameters. The classification performance was evaluated using the F_1 measure for the single-argument classifiers and the accuracy for the multiclassifiers.

6.2 Automatic VerbNet class vs. automatic FrameNet frame detection

In these experiments, we classify the ILCs on PB and FN and the frames on FN. For the training stage we use SVMs with Tree Kernels.

For ILC detection the results are depicted in Table 2. The first six columns report the F_1 measure of some verb class classifiers whereas the last column shows the global multiclassifier accuracy. We note the ILC results on PB are similar to those obtained for the ILCs on FN. This suggests that the training corpus does not have a major influence. Also, the SCF-based tree kernel seems to be robust with respect to the quality of the parse trees. The performance decay is very small on FN that uses automatic parse trees with respect to PB that contains gold parse trees.

For frame detection on FN, we trained our classifier on 46,734 training instances and tested on 11,650 testing instances, obtaining an accuracy of 91.11%. Consequently, the ILC detection on both PB (92.96%) and FN (92.63%) is more accurate than the frame detection.

6.3 Automatic semantic role labeling on FrameNet

In the experiments involving semantic role labeling, we used SVMs with polynomial kernels. We adopted the standard features developed for semantic role detection by Gildea and Jurafsky : *Predicate*, *Headword*, *Phrase Type*, *Governing Category*, *Position*, *Voice* and *Path*. Also, we considered some of the features designed by [20]: *First and Last Word/POS in Constituent*, *Subcategorization*, *Head Word of Prepositional Phrases* and the *Syntactic Frame* feature from [22]. For the rest of the paper, we will refer to these features as being literature features (LF). The results obtained when using the literature features alone or in conjunction with the gold frame feature, gold ILC, automatically detected frame feature and automatically detected ILC are depicted in Table 3.

The first four columns report the F_1 measure of some role classifiers whereas the last column shows the global multiclassifier accuracy. The first row contains the number of training and testing instances and each of the other rows contains the performance obtained for different feature combinations. The results are reported for the labeling task as the argument-boundary detection task is not affected by the frame-like features [6].

We note that automatic frame produces an accuracy very close to the one obtained with automatic ILCs suggesting that these latter are very good candidate for replacing the frame feature. Also, both automatic features are very effective, decreasing the error rate of 20%.

To test the impact of the ILC feature on SRL with different amount of training data, we additionally draw the learning curves with respect to different features: LF, LF+ gold ILC , LF+automatic ILC trained on PB and LF+automatic ILC trained on FN. As can be noted, the automatic ILC information provided by the ILC classifiers (trained on FN or PB) performs almost as good as the gold ILC (Figure 4).

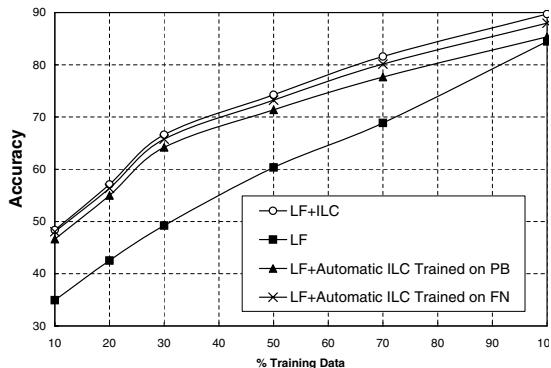


Figure 4. Semantic role learning curve.

6.4 Annotating PB with FN semantic roles

To show that our approach can be suitable for semantic role free-text annotation, we have automatically classified PB sentences with the FN semantic-role classifiers. In order to measure the quality of the annotation, we randomly selected 100 sentences and manually verified them. We measured the performance obtained with and without the automatic ILC feature. The sentences contained 189 arguments from which 35 were incorrect when ILC was used compared to 72 incorrect in the absence of this feature, i.e. an accuracy of 81% with ILC versus 62% without it. This demonstrates the importance of the ILC feature outside the scope of FrameNet where the frame feature is not available.

7 CONCLUSIONS

In this paper, we pursue Levin's thesis and we automatically classify verbs based on their predicate-argument structure and their selectional preferences on different argument slots. We show that Tree Kernels and structural features are more suitable for our goal compared with other methods that use for example linear features. Also, by comparing our structural features with previously developed and linguistically motivated features we found a reasonably degree of overlap. Additionally, we demonstrate that our approach is very robust and can be applied successfully on any type of parse trees (gold or automatic) or corpora.

Regarding the impact of ILC feature on SRL, we have shown that it can replace the frame feature without appreciable accuracy decay. This is very important as (1) several frames are not covered by enough training data and (2) thanks to our mapping algorithm we can reuse the data from PropBank to extend the verb coverage.

REFERENCES

- [1] Eugene Charniak, 'A maximum-entropy-inspired parser', in *NACL00*, Seattle, Washington, (2000).
- [2] Hoa Trang Dang, Karin Kipper, Martha Palmer, and Joseph Rosenzweig, 'Investigating regular sense extensions based on interseective levin classes', in *COLING-ACL98*, (1998).
- [3] Bonnie Dorr, 'Large-scale dictionary construction for foreign language tutoring and interlingual machine translation', *Machine Translation*, (1997).
- [4] Bonnie Dorr and Doug Jones, 'Role of word sense disambiguation in lexical acquisition: Predicting semantics from syntactic cues', in *COLING-ACL96*, (1996).
- [5] Charles J. Fillmore, 'The case for case', in *Universals in Linguistic Theory*, (1968).
- [6] Daniel Gildea and Daniel Jurafsky, 'Automatic labeling of semantic roles', *Computational Linguistics*, (2002).
- [7] Ana-Maria Giuglea and Alessandro Moschitti, 'Semantic role labeling via FrameNet, VerbNet and PropBank', in *COLING-ACL06*, Sydney, Australia, (2006).
- [8] Sabine Schulte im Walde, 'Clustering verbs semantically according to their alternation behaviour', *COLING*, (2000).
- [9] T. Joachims, 'Making large-scale SVM learning practical.', in *Advances in Kernel Methods - Support Vector Learning*, eds., B. Schölkopf, C. Burges, and A. Smola, (1999).
- [10] Christopher Johnson, Miriam Petrucc, Collin Baker, Michael Ellsworth, Josef Ruppenhofer, and Charles Fillmore, 'Framenet: Theory and practice', *Berkeley, California*, (2003).
- [11] Paul Kingsbury and Martha Palmer, 'From Treebank to PropBank', in *LREC02*, (2002).
- [12] Karin Kipper, Hoa Trang Dang, and Martha Palmer, 'Class-based construction of a verb lexicon', in *AAA100*, (2000).
- [13] Mirella Lapata and Chris Brew, 'Using subcategorization to resolve verb class ambiguity.', in *Joint SIGDAT Conference on EMNLP and Very Large Corpora*, eds., Pascal Fung and Joe Zhou, College Park, Maryland, (1999).
- [14] Mirella Lapata and Chris Brew, 'Verb class disambiguation using informative priors.', *Computational Linguistics*, (2004).
- [15] Beth Levin, *English Verb Classes and Alternations A Preliminary Investigation*, Chicago: University of Chicago Press., 1993.
- [16] Gina-Anne Levow, Bonnie Dorr, and Dekang Lin, 'Construction of chinese-english semantic hierarchy for information retrieval', *Technical report,University of Maryland, College Park*, (2000).
- [17] Kenneth Litkowski, 'Senseval-3 task automatic labeling of semantic roles', in *Senseval-3*, (2004).
- [18] Paola Merlo and Suzanne Stevenson, 'Automatic verb classification based on statistical distribution of argument structure', *Computational Linguistics*, (2001).
- [19] Alessandro Moschitti, 'A study on convolution kernels for shallow semantic parsing', in *ACL04*, Barcelona, Spain, (2004).
- [20] Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky, 'Support vector learning for semantic argument classification', *Machine Learning*, (2005).
- [21] Suzanne Stevenson, 'Automatic verb classification using grammatical features', in *EACL99*, Bergen, Norway, (1999).
- [22] Nianwen Xue and Martha Palmer, 'Calibrating features for semantic role labeling', in *Proceedings of EMNLP 2004*, Barcelona, Spain, (2004). ACL04.

Semantic Tree Kernels to classify Predicate Argument Structures

Alessandro Moschitti¹ and Bonaventura Coppola² and Daniele Pighin³ and Roberto Basili⁴

Abstract. Recent work on Semantic Role Labeling (SRL) has shown that syntactic information is critical to detect and extract predicate argument structures. As syntax is expressed by means of structured data, i.e. parse trees, its encoding in learning algorithms is rather complex.

In this paper, we apply tree kernels to encode the whole predicate argument structure in Support Vector Machines (SVMs). We extract from the sentence syntactic parse the subtrees that span potential argument structures of the target predicate and classify them in incorrect or correct structures by means of tree kernel based SVMs. Experiments on the PropBank collection show that the classification accuracy of correct/incorrect structures is remarkably high and helps to improve the accuracy of the SRL task. This is a piece of evidence that tree kernels provide a powerful mechanism to learn the complex relation between syntax and semantics.

1 INTRODUCTION

The design of features for natural language processing tasks is, in general, a critical problem. The inherent complexity of linguistic phenomena, often characterized by structured data, makes difficult to find effective attribute-value representations for the target learning models.

In many cases, the traditional feature selection techniques [8] are not very useful since the critical problem relates to feature generation rather than selection. For example, the design of features for a natural language syntactic parse-tree re-ranking problem [2] cannot be carried out without a deep knowledge about automatic syntactic parsing. The modeling of syntax/semantics-based features should take into account linguistic aspects to detect the interesting context, e.g. ancestor nodes or semantic dependencies [15].

A viable alternative has been proposed in [3], where convolution kernels were used to implicitly define a tree substructure space. The selection of the relevant structural features was left to the voted perceptron learning algorithm. Such successful experimentation shows that tree kernels are very promising for automatic feature engineering, especially when the available knowledge about the phenomenon is limited.

In a similar way, automatic learning tasks that rely on syntactic information may take advantage of a tree kernel approach. One of such tasks is the Semantic Role Labeling (SRL), as defined e.g. in [1] over the PropBank corpus [7]. Most literature work models SRL as the classification of tree nodes of the sentence parse containing the target predicate. Indeed, a node can uniquely determine the set of words that compose an argument (boundaries) and provide, along

with the local tree structure, information useful to the classification of the role. Accordingly, most SRL systems split the labeling process into two different steps: Boundary Detection (i.e. determine the text boundaries of predicate arguments) and Role Classification (i.e. labeling such arguments with a semantic role, e.g. Arg0 or Arg1).

Both the above steps require the design and extraction of features from the parse tree. Capturing the interconnected relationships among a predicate and its arguments is a hard task. To decrease such complexity we can design features considering a predicate with only one argument at a time, but this limits our ability to capture the semantics of the whole predicate structure. An alternative approach to engineer syntactic features is the use of tree kernels as the substructures that they generate potentially correspond to relevant syntactic clues.

In this paper we use tree kernels to model classifiers that decide if a predicate argument structure is correct or not. We apply a traditional boundary classifier (*TBC*) [11] to label all parse tree nodes that are potential arguments, then we classify the syntactic subtrees which span the predicate-argument dependencies, i.e. Predicate Argument Spanning Trees (*PASTs*). Since the design of effective features to encode such information is not simple, tree kernels are a very useful method. To validate our approach we experimented tree kernels with Support Vector Machines for the classification of *PASTs*. The results show that this classification problem can be learned with high accuracy (about 88% of F_1 -measure⁵) and the impact on the overall SRL labeling accuracy is also relevant.

The paper is organized as follows: Section 2 introduces the Semantic Role Labeling based on SVMs and the tree kernel spaces; Section 3 formally defines the *PASTs* and the algorithm to classify them; Section 4 shows the comparative results between our approach and the traditional one; Section 5 presents the related work; and finally, Section 6 summarizes the conclusions.

2 SEMANTIC ROLE LABELING

In the last years, several machine learning approaches have been developed for automatic role labeling, e.g. [5, 11]. Their common characteristic is the adoption of attribute-value representations for predicate-argument structures. Accordingly, our basic system is similar to the one proposed in [11] and is hereby described.

We use a boundary detection classifier (for any role type) to derive the words compounding an argument and a multiclassifier to assign the role (e.g. ARG0 or ARG1) described in PropBank [7]). To prepare the training data for both classifiers, we used the following algorithm:

- Given a sentence from the *training-set*, generate a full syntactic parse tree;

⁵ F_1 assigns equal importance to Precision P and Recall R , i.e. $F_1 = \frac{2P \times R}{P + R}$.

¹ University of Rome “Tor Vergata”, moschitti@info.uniroma2.it

² ITC-Irst and University of Trento, coppolab@itc.it

³ University of Rome “Tor Vergata”, daniele.pighin@gmail.com

⁴ University of Rome “Tor Vergata”, basili@info.uniroma2.it

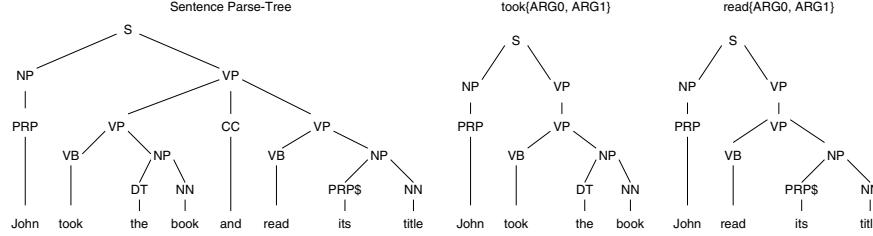


Figure 1. A sentence parse tree with two predicative subtree structures (*PASTs*)

2. Let P and A be respectively the set of predicates and the set of parse-tree nodes;

3. For each pair $\langle p, a \rangle \in P \times A$:

- extract the feature representation set, $F_{p,a}$;
- if the subtree rooted in a covers exactly the words of one argument of p , put $F_{p,a}$ in the T^+ set (positive examples), otherwise put it in the T^- set (negative examples).

The outputs of the above algorithm are the T^+ and T^- sets. For the subtask of Boundary Detection these can be directly used to train a boundary classifier (e.g. an SVM). Concerning the subtask of Role Classification, the generic binary role labeler for role r (e.g. an SVM) can be trained on the T_r^+ , i.e. its positive examples and T_r^- , i.e. its negative examples, where $T^+ = T_r^+ \cup T_r^-$, according to the ONE-vs-ALL scheme. The binary classifiers are then used to build a general role multiclassifier by simply selecting the argument associated with the maximum among the classification scores resulting from the individual binary SVM classifiers.

Regarding the design of features for predicate-argument pairs, we can use the attribute-values defined in [5] or tree structures [10]. Although we focus on the latter approach, a short description of the former is still relevant as they are used by *TBC*. They include the *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* features. For example, the *Phrase Type* indicates the syntactic type of the phrase labeled as a predicate argument and the *Parse Tree Path* contains the path in the parse tree between the predicate and the argument phrase, expressed as a sequence of nonterminal labels linked by direction (up or down) symbols, e.g. $V \uparrow VP \downarrow NP$.

A viable alternative to manual design of syntactic features is the use of tree-kernel functions. These implicitly define a feature space based on all possible tree substructures. Given two trees T_1 and T_2 , instead of representing them with the whole fragment space, we can apply the kernel function to evaluate the number of common fragments.

Formally, given a tree fragment space $\mathcal{F} = \{f_1, f_2, \dots, f_{|\mathcal{F}|}\}$, the indicator function $I_i(n)$ is defined, which is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1 's and T_2 's nodes, respectively. In turn $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{l(f_i)} I_i(n_1) I_i(n_2)$, where $0 \leq \lambda \leq 1$ and $l(f_i)$ is the number of levels of the subtree f_i . Thus $\lambda^{l(f_i)}$ assigns a lower weight to larger fragments. When $\lambda = 1$, Δ is equal to the number of common fragments rooted at nodes n_1 and n_2 . As described in [3], Δ can be computed in $O(|N_{T_1}| \times |N_{T_2}|)$.

3 AUTOMATIC CLASSIFICATION OF PREDICATE ARGUMENT STRUCTURES

Most semantic role labeling models rely only on the features extracted from the current candidate argument node. To consider a complete predicate argument structure, the classifier should formulate a

hypothesis on the potential parse-tree node subsets which include the argument nodes of the target predicate. Without the boundary information, we should consider all possible tree node subsets, i.e. an exponential number.

To solve such problems we apply a traditional boundary classifier (*TBC*) to select the set of potential arguments \mathcal{PA} . Such a subset can be associated with a subtree which in turn can be classified by means of a tree kernel function. Intuitively, such a function measures to what extent a given candidate subtree is *compatible* with the subtree of a correct predicate argument structure.

3.1 The Predicate Argument Spanning Trees (*PASTs*)

We consider the predicate argument structures annotated in PropBank along with the corresponding TreeBank data as our object space. Given the target predicate p in a sentence parse tree T and a subset $s = \{n_1, \dots, n_k\}$ of its nodes, N_T , we define as the spanning tree root r the lowest common ancestor of n_1, \dots, n_k . The node set spanning tree p_s is the subtree rooted in r from which the nodes that are neither ancestors nor descendants of any n_i are removed.

Since predicate arguments are associated with tree nodes (i.e. they exactly fit into syntactic constituents), we can define the *Predicate Argument Spanning Tree (PAST)* of a predicate argument set, $\{a_1, \dots, a_n\}$, as the node set spanning tree (*NST*) over such nodes, i.e. $p_{\{a_1, \dots, a_n\}}$. A *PAST* corresponds to the *minimal* subparse tree whose leaves are all and only the words compounding the arguments. For example, Figure 1 shows the parse tree of the sentence "John took the book and read its title". $took_{\{ARG_0, ARG_1\}}$ and $read_{\{ARG_0, ARG_1\}}$ are two *PAST* structures associated with the two predicates *took* and *read*, respectively. All the other possible *NSTs* are not valid *PASTs* for these predicates. Note that labeling $p_s, \forall s \subseteq N_T$ with a *PAST* Classifier is equivalent to solve the boundary detection problem.

The critical points for the application of *PASTs* are: (1) how to design suitable features for the characterization of *PASTs*. This new structure requires a careful linguistic investigation about its significant properties. (2) How to deal with the exponential number of *NSTs*.

For the first problem, the use of tree kernels over the *PASTs* can be an alternative to the manual feature design as the learning machine, (e.g. SVMs) can select the most relevant features from a high dimensional feature space. In other words, we can use a tree kernel function to estimate the similarity between two *PASTs* (see Section 2), hence avoiding to define explicit features.

For the second problem there are two main approaches: (1) We can consider the classification confidence provided by *TBC* [11] and evaluate the m most probable argument node sequences $\{n_1, \dots, n_k\}$. On the m *NSTs* derived from such sequences, we can apply a re-ranking approach based on SVMs with tree kernel. (2) We can use only the set of nodes \mathcal{PA} decided by *TBC* (i.e. those classified as

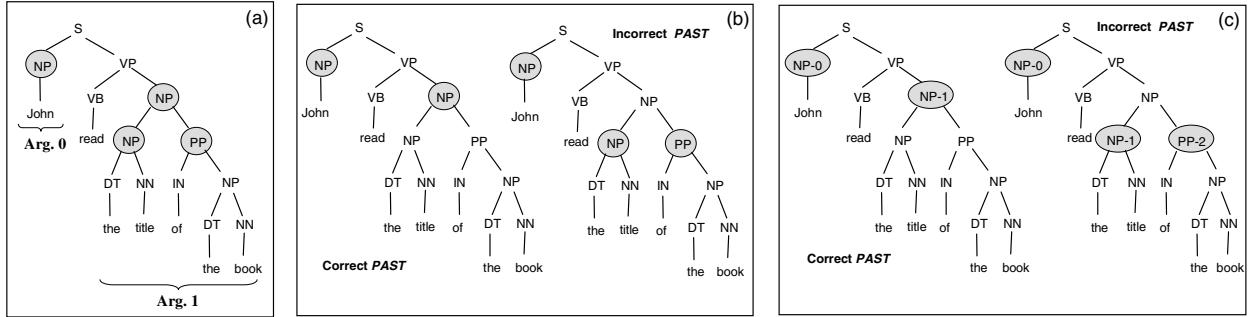


Figure 2. Two-step boundary classification. a) Sentence tree; b) Two candidate *PAST*s; c) Extended *PAST*-*Ord* labeling

arguments). Thus we need to classify only the set \mathcal{P} of *NSTs* associated with any subset of \mathcal{PA} , i.e. $\mathcal{P} = \{p_s : s \subseteq \mathcal{PA}\}$.

As a re-ranking task would not give an explicit and clear indication of the classifier ability to distinguish between correct and incorrect predicate argument structures, we preferred to apply the second approach. However, also the classification of \mathcal{P} may be computationally problematic, since theoretically there are $|\mathcal{P}| = 2^{|\mathcal{PA}|}$ members.

In order to develop a very efficient procedure, we applied the *PAST* Classifier only to structures that we know that are incorrect. A simple way to detect them is to look for node pairs $\langle n_1, n_2 \rangle \in \mathcal{PA} \times \mathcal{PA}$ that overlap, i.e. either n_1 is ancestor of n_2 or vice versa. Note that structures that contain overlapping nodes often also contain correct substructures, i.e. subsets of \mathcal{PA} can be associated with correct *PAST*. Assuming the above hypothesis, we create two node sets $\mathcal{PA}_1 = \mathcal{PA} - \{n_1\}$ and $\mathcal{PA}_2 = \mathcal{PA} - \{n_2\}$ and classify them with the *PAST* Classifier to select the correct set of argument boundaries. This procedure can be generalized to a set of overlapping nodes greater than 2 by selecting a maximal set of non-overlapping nodes. Additionally, as the Precision of *TBC* is generally high, the number of overlapping nodes is very small. Thus we can explore the whole space.

Figure 2 shows a working example of the multi-stage classifier. In Frame (a), *TBC* labels as potential arguments (gray color) three overlapping nodes related to ARG1. This leads to two possible solutions (Frame (b)) of which only the first is correct. In fact, according to the second one, the propositional phrase *of the book* would be incorrectly attached to the verbal predicate, i.e. in contrast with the parse tree. The *PAST* Classifier, applied to the two *NSTs*, is expected to detect this inconsistency and provide the correct output.

3.2 Designing Features with Tree Fragments

The Frame (b) of Figure 2 shows two perfectly identical *NSTs*. Therefore, it is not possible to discern between them using only their fragments. To solve the problem we can enrich the *NSTs* by marking their argument nodes with a progressive number, starting from the leftmost argument. For example, in the first *NST* of Frame (c), we mark as NP-0 and NP-1 the first and second argument nodes whereas in the second *NST* we transform the three argument node labels in NP-0, NP-1 and PP-2. We will refer to the resulting structure as a *PAST*-*Ord* (ordinal number). This simple modification allows the tree kernel to generate different argument structures for the above *NSTs*. For example, from the first *NST* in Figure 2.c, the fragments [NP-1 [NP] [PP]], [NP [DT] [NN]] and [PP [IN] [NP]] are generated. They do not match anymore with the [NP-0 [NP] [PP]], [NP-1 [DT] [NN]] and [PP-2 [IN] [NP]] fragments generated from the second *NST* in Figure

2.c.

We also explored another relevant direction in enriching the feature space. It should be noted that the semantic information provided by the role type can remarkably help the detection of correct or incorrect predicate argument structures. Thus, we enrich the argument node label with the role type, e.g. the NP-0 and NP-1 of the correct *PAST* of Figure 2.c becomes NP-Arg0 and NP-Arg1 (not shown). We refer to this structure as *PAST*-*Arg*. Of course, to apply the *PAST*-*Arg* Classifier, we need a traditional role multiclassifier (*TRM*) which labels the arguments detected by *TBC*.

4 THE EXPERIMENTS

The experiments were carried out within the setting defined in the CoNLL-2005 Shared Task [1]. We used the PropBank corpus available at www.cis.upenn.edu/~ace, along with the Penn TreeBank 2 for the gold trees (www.cis.upenn.edu/~treebank) [9], which includes about 53,700 sentences.

Since the experiments over gold parse trees inherently overestimate the accuracy in the semantic role labeling task, e.g. 93% vs. 79% [11], we also adopted Charniak parse trees from the CoNLL 2005 Shared Task data (available at www.lsi.upc.edu/~srlconll/) along with the official performance evaluator.

All the experiments were performed with the SVM-light software [6] available at svmlight.joachims.org. For *TBC* and *TRM*, we used the linear kernel with a regularization parameter (option *-c*) equal to 1. A cost factor (option *-j*) of 10 was adopted for *TBC* to have a higher Recall, whereas for *TRM*, the cost factor was parameterized according to the maximal accuracy of each argument class on the validation set. For the *PAST* Classifier, we implemented the tree kernel defined [3] inside SVM-light with a λ equal to 0.4 (see [10]).

4.1 Gold Standard Tree Evaluations

In these experiments, we used the sections from 02 to 08 of the TreeBank/PropBank (54,443 argument nodes and 1,343,046 non-argument nodes) to train the traditional boundary classifier (*TBC*). Then, we applied it to classify the sections from 09 to 21 (125,443 argument nodes vs. 3,010,673 non-argument nodes). We obtained 2,988 *NSTs* containing at least one overlapping node pair out of the total 65,212 predicate structures (according to the *TBC* decisions). From the 2,988 overlapping structures, we derived 3,624 positive and 4,461 negative *NSTs*, that we used to train the *PAST*-*Ord* Classifier.

The performance was evaluated through the F_1 measure over Section 23, which includes 10,406 argument nodes out of 249,879 parse

	TBC			TBC+RND			TBC+HEU			TBC+PAST-Ord		
	P.	R.	F_1	P.	R.	F_1	P.	R.	F_1	P.	R.	F_1
All Struct.	92.2	98.8	95.4	93.6	97.3	95.4	93.0	97.3	95.1	94.4	98.4	96.4
Overl. Struct.	98.3	65.8	78.8	74.0	72.3	73.1	68.1	75.2	71.5	89.6	92.7	91.1

Table 1. Two-step boundary classification performance using the *TBC*, *RND* and *HEU* baselines, and the *PAST*-Ord classifier.

	Section 21						Section 23					
	bnd			bnd+class			bnd			bnd+class		
	PAST Classifier		<i>RND</i>									
	-	Ord	Arg									
P.	87.5	88.3	88.3	86.9	85.5	86.3	86.4	85.0	78.6	79.0	79.3	77.8
R.	87.3	88.1	88.3	87.1	85.7	86.5	86.8	85.6	78.1	78.4	78.7	77.9
F_1	87.4	88.2	88.3	87.0	85.6	86.4	86.6	85.3	78.3	78.7	79.0	77.9
									73.4	73.8	73.9	72.9

Table 2. Semantic Role Labeling performance on automatic trees using *PAST* classifiers.

tree nodes. After applying the *TBC* classifier, we detected 235 overlapping *NSTs*, from which we extracted 204 correct *PASTs* and 385 incorrect ones. On such gold standard trees, we measured only the performance of the *PAST*-Arg Classifier which was very high, i.e. 87.1% in Precision and 89.2% in Recall (88.1% of F_1).

Using the *PAST*-Ord Classifier we removed from the *TBC* outcome the nodes that caused overlaps. To measure the impact on the boundary detection task, we compared it with three different boundary classification baselines:

1. *TBC*: overlaps are ignored and no decision is taken. This provides an upper bound for the recall as no potential argument is rejected for later labeling. Notice that, in presence of overlapping nodes, the sentence cannot be annotated correctly.

2. *RND*: one among the non-overlapping structures with maximal number of arguments is randomly selected.

3. *HEU* (heuristic): one of the *NSTs* which contains the nodes with the lowest overlapping score is chosen. This score counts the number of overlapping node pairs in the *NST*. For example, in Figure 2.a we have an NP that overlaps with two nodes NP and PP, thus it is assigned a score of 2.

The third row of Table 1 shows the results of *TBC*, *TBC+RND*, *TBC+HEU* and *TBC+PAST-Ord* in the columns 2,3,4 and 5, respectively. We note that: First, the *TBC* F_1 is slightly higher than the result obtained in [11], i.e. 95.4% vs. 93.8% under the same training/testing conditions (i.e. same PropBank version, same training and testing split and same machine learning algorithm). This is explained by the fact that we did not include continuations and co-referring arguments that are more difficult to detect. Second, both *RND* and *HEU* do not improve the *TBC* result. This can be explained by observing that in the 50% of the cases a correct node is removed. Third, when the *PAST*-Ord Classifier is used to select the correct node, the F_1 increases of 1.49%, i.e. (96.86 vs. 95.37). This is a relevant result as it is difficult to increase the very high baseline given by *TBC*. Finally, we tested the above classifiers on the overlapping structures only, i.e. we measured the *PAST*-Ord Classifier improvement on all and only the structures that required its application. Such reduced test set contains 642 argument nodes and 15,408 non-argument nodes. The fourth row of Table 1 reports the classifier performance on such task. We note that the *PAST*-Ord Classifier improves the other heuristics of about 20%.

4.2 Automatic Tree Evaluations

In these experiments we used the automatic trees generated by Charniak's parser and the predicate argument annotations defined in the

CoNLL 2005 shared task. Again, we trained *TBC* on sections 02-08 whereas, to achieve a very accurate role classifier, we trained *TRM* on all sections 02-21. Then, we trained the *PAST*, *PAST*-Ord, and *PAST*-Arg Classifiers on the output of *TBC* and *TRM* over sections 09-20 for a total of 183,642 arguments, 30,220 *PASTs* and 28,143 incorrect *PASTs*.

PAST Class.	Section 21			Section 23		
	P.	R.	F_1	P.	R.	F_1
—	69.8	77.9	73.7	62.2	77.1	68.9
Ord	73.7	81.2	77.3	63.7	80.6	71.2
Arg	73.6	84.7	78.7	64.2	82.3	72.1

Table 3. *PAST*, *PAST*-Ord, and *PAST*-Arg performances on sections 21 and 23.

First, to test the *TBC*, *TRM* and the *PAST* classifiers, we used Section 23 (17,429 arguments, 2,159 *PASTs* and 3,461 incorrect *PASTs*) and Section 21 (12,495 arguments, 1,975 *PASTs* and 2,220 incorrect *PASTs*). The performance derived on Section 21 corresponds to an upper bound of our classifiers, i.e. the results using an ideal syntactic parser (Charniak's parser was trained also on Section 21) and an ideal role classifier. They provide the *PAST* family classifiers with accurate syntactic and semantic information. Table 3 shows Precision, Recall and F_1 measures of the *PAST* classifiers over the *NSTs* of sections 21 and 23. Rows 2, 3 and 4 report the performance of *PAST*, *PAST*-Ord, and *PAST*-Arg Classifiers, respectively. Several points should be remarked: (a) the general performance is lower than the one achieved on gold trees with *PAST*-Ord, i.e. 88.1% (see Section 4.1). The impact of parsing accuracy is also confirmed by the gap of about 6% points between sections 21 and 23. (b) The ordinal numbering of arguments (*Ord*) and the role type information (*Arg*) provide the tree kernel with more meaningful fragments since they improve the basic model of about 4%. (c) The deeper semantic information generated by the *Arg* labels provides useful clues to select correct predicate argument structures, since it improves the *Ord* model on both sections.

Second, we measured the impact of the *PAST* classifiers on both phases of semantic role labeling. Table 2 reports the results on the sections 21 and 23. For each of them, the Precision, Recall and F_1 of different approaches to the boundary identification (bnd) and to the complete task, i.e. boundary and role classification (bnd+class), is shown. Such approaches are based on different strategies to remove the overlaps, i.e. *PAST*, *PAST*-Ord, *PAST*-Arg and the baseline (*RND*) which uses a random selection of non-overlapping struc-

tures. We needed to remove the overlaps from the baseline in order to apply the CoNLL evaluator.

We note that: (a) for any model, the boundary detection F_1 on Section 21 is about 10 points higher than the F_1 on Section 23 (e.g. 87.0% vs. 77.9% for *RND*). As expected, the parse tree quality is very important to detect argument boundaries. (b) On the real test (Section 23) the classification introduces labeling errors which decrease the accuracy of about 5% (77.9 vs 72.9 for *RND*). (c) The *Ord* and *Arg* approaches constantly improve the baseline F_1 of about 1%. Such a result does not surprise as it is similar to the one obtained on Gold Trees: the overlapping structures are a small percentage of the test set thus the overall impact cannot be very high.

Third, the comparison with the CoNLL 2005 results [1] can only be carried out with respect to the whole SRL task (bnd+class in table 2) since boundary detection versus role classification is generally not provided in CoNLL 2005. Moreover, our best global result, i.e. 73.9%, was obtained under two severe experimental factors: a) the use of just 1/3 of the available training data, and b) the usage of the linear SVM model for the TBC classifier, which is much faster than the polynomial SVMs but also less accurate. However, we note the promising results of the *PAST* meta-classifier, which can be used with any of the best figure CoNLL systems.

Finally, kernel outcome suggests that: (a) it is robust to parse tree errors since it preserves the same improvement across trees derived with different accuracy, i.e. the gold trees of Penn TreeBank and the automatic trees of Section 21 and Section 23. (b) It shows a high accuracy for the classification of correct and incorrect predicate argument structures. This last property is quite interesting considering the important findings of a recent paper [13]. The winning strategy to improve semantic role labeling relates to the exploiting of different labeling hypotheses, i.e. several \mathcal{PA}_i sets derived from different parsing alternatives. A joint inference procedure was used to select the most likely set $s \subseteq \cup_i \mathcal{PA}_i$. In our opinion, the *PAST* Classifiers seem very well suited to select such set.

5 RELATED WORK

Recently, many kernels for natural language applications have been designed. In what follows, we highlight their difference and properties.

The tree kernel used in this article was proposed in [3] for syntactic parsing reranking. It was experimented with the Voted Perceptron and was shown to improve the syntactic parsing. In [4], a feature description language was used to extract structural features from the syntactic shallow parse trees associated with named entities. The experiments on the named entity categorization showed that when the description language selects an adequate set of tree fragments the Voted Perceptron algorithm increases its classification accuracy. The explanation was that the complete tree fragment set contains many irrelevant features and may cause overfitting. In [13], a set of different syntactic parse trees, e.g. Charniak n best parse trees, were used to improve the SRL accuracy. These different sources of syntactic information were used to generate a set of different SRL outputs. A joint inference stage was applied to resolve the inconsistency of the different outputs. This approach may be applied to our tree kernel strategies to design a joint tree kernel model. In [14], it was observed that there are strong dependencies among the labels of the semantic argument nodes of a verb. Thus, to approach the problem as the classification of an overall role sequences, a re-ranking method is applied to the assignments generated by a *TRM*. This approach is in line with our *PAST* Classifier that can be used to refine such re-ranking strategy. In [12], some experiments were conducted on SRL

systems trained using different syntactic views. Again, our approach may be used in conjunction with this model to provide a further syntactic view related to the whole predicate argument structure.

Acknowledgments

This research is partially supported by the PrestoSpace EU Project#: FP6-507336.

6 CONCLUSIONS

The feature design for new natural language learning tasks is difficult. We can take advantage of the kernel methods to model our intuitive knowledge about the target linguistic phenomenon. In this paper we have shown that we can exploit the properties of tree kernels to engineer syntactic features for the semantic role labeling task.

The experiments on gold standard trees as well as on automatic trees suggest that (1) the information related to the whole predicate argument structure is important and (2) tree kernels can be used to generate syntactic/semantic features. The remarkable result is that such structures are robust with respect to parse tree errors.

In the future, we would like to use an approach similar to the *PAST* classifier to select the best predicate argument annotation from those carried out on several parse trees provided by one or more parsing models.

REFERENCES

- [1] Xavier Carreras and Lluís Márquez, ‘Introduction to the CoNLL-2005 shared task: Semantic role labeling’, in *Proceedings of CoNLL-2005*, (2005).
- [2] Michael Collins, ‘Discriminative reranking for natural language parsing’, in *In Proceedings of ICML 2000*, (2000).
- [3] Michael Collins and Nigel Duffy, ‘New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron’, in *ACL02*, (2002).
- [4] Chad Cumby and Dan Roth, ‘Kernel methods for relational learning’, in *Proceedings of ICML 2003*, Washington, DC, USA, (2003).
- [5] Daniel Gildea and Daniel Jurafsky, ‘Automatic labeling of semantic roles’, *Computational Linguistic*, **28**(3), 496–530, (2002).
- [6] T. Joachims, ‘Making large-scale SVM learning practical.’, in *Advances in Kernel Methods - Support Vector Learning*, eds., B. Schölkopf, C. Burges, and A. Smola, (1999).
- [7] Paul Kingsbury and Martha Palmer, ‘From Treebank to PropBank’, in *Proceedings of LREC’02*, Las Palmas, Spain, (2002).
- [8] Ron Kohavi and Dan Sommerfield, ‘Feature subset selection using the wrapper model: Overfitting and dynamic search space topology’, in *1st KDD Conference*, (1995).
- [9] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, ‘Building a large annotated corpus of english: The Penn Treebank’, *Computational Linguistics*, **19**, 313–330, (1993).
- [10] Alessandro Moschitti, ‘A study on convolution kernel for shallow semantic parsing’, in *Proceedings of ACL’04*, Barcelona, Spain, (2004).
- [11] Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky, ‘Support vector learning for semantic argument classification’, *Machine Learning Journal*, (2005).
- [12] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky, ‘Semantic role labeling using different syntactic views’, in *Proceedings of ACL’05*, (2005).
- [13] V. Punyakanok, D. Roth, and W. Yih, ‘The necessity of syntactic parsing for semantic role labeling’, in *Proceedings of IJCAI 2005*, (2005).
- [14] Kristina Toutanova, Aria Haghighi, and Christopher Manning, ‘Joint learning improves semantic role labeling’, in *Proceedings of ACL’05*, (2005).
- [15] Kristina Toutanova, Penka Markova, and Christopher D. Manning, ‘The leaf projection path view of parse trees: Exploring string kernels for hpsg parse selection’, in *In Proceedings of EMNLP 2004*, (2004).
- [16] Nianwen Xue and Martha Palmer, ‘Calibrating features for semantic role labeling’, in *Proceedings of EMNLP 2004*, (2004).

7. Planning and Scheduling

This page intentionally left blank

A Multivalued logic model of planning

M. Baioletti and A. Milani and V. Poggioni and S. Suriani¹

Abstract. In this work a model for planning with multivalued fluents and graded actions, based on the infinite valued Łukasiewicz logic, is introduced. In multivalued planning, fluents can assume truth values in the interval [0, 1] and actions can be executed at different application degrees also varying in [0, 1]. The notions of planning problem and solution plan also reflect a multivalued approach. Multivalued fluents and graded actions allow to model many real situations where some features of the world cannot be modeled with boolean values and where actions can be executed with varying strength which produces graded effects as well. Even if most existing planning models fail to address this kind of domains, our model is comparable with models allowing flexible actions and soft constraints. A correct/complete algorithm which solves bounded multivalued planning problems based on MIP compilation is also described and a prototype implementation is presented.

1 Introduction

Many-valued logics have recently been considered in Software Engineering research because they provide a support for an explicit modelling of uncertainty, disagreement and fuzziness. In many areas of artificial intelligence research, related to AI planning, several extensions of classical models with multivalued approaches have been proposed. In model checking, the classical approach has been extended to a multi-valued framework [4]. In [2] a new framework for soft constraint satisfaction problems (CSP), in which fuzzy CSP and weighted CSP are particular cases, has been proposed. In [10] a many-valued SAT solver based on clauses with multivalued literals has been presented.

In AI planning there exist many real application domains in which the relevant properties cannot be expressed by a crisp boolean value, the actions can be applied with a graded level and the effects can depend on that level. Consider for instance a simple scenario where the action *open_door* is available in the domain and modifies the truth value of the fluent *door_is_open*.

A typical probabilistic planning model would treat both the fluent *door_is_open* and the successful execution of the action *open_door* as uncertain, assigning to them a belief degree in [0, 1]. Such degrees are different from the truth value that the real world will assign to the fluent and to the action, which will be definitely 1 or 0, i.e. true/false or successful/unsuccessful. In other words, the planner is uncertain about boolean, but unknown, properties like *door_is_open*. A planner with resources would treat the fluent *door_is_open* as a real function whose initial value has to be measured, for instance by the openness angle. Since the action *open_door* cannot have as a numerical parameter the angle by which *door_is_open* is changed, it is not possible to model the strength of the action. From a multivalued point

of view, instead, the truth value of the fluent *door_is_open* denotes a real world property which can vary from 0 (the door is completely closed) to 1 (the door is completely opened). In other words the door can “really” be opened at an intermediate degree, between “completely closed” and “completely opened”. Moreover the multivalued action *open_door* can be applied with a “strength” ranging from 0 (not applied at all) to 1 (completely applied). The resulting effect on the fluent *door_is_open* will depend on the application level of action *open_door* and on the previous truth value of the fluent itself, i.e. the effects can also be additive.

Choosing a suitable execution degree of an action *a*, it is possible to obtain a sufficient level for some fluents, for instance goals or subgoals, while keeping low the side effects of the execution of *a*.

The use of graded actions is also useful in domains where the execution degree is related to the cost/execution time of the action. Therefore it is possible to define a metric function based on execution degrees by which optimal plans in terms of overall cost/time are generated.

It is worth noticing that many fuzzy concepts cannot be represented by numerical resources, because the fuzzy values could not be obtainable by a measurement process or they could be affected by subjective factors.

In the multivalued/fuzzy logics scenario we preferred to use a logic based on T-norms because of its well-founded mathematical aspects. Among this class of logics we have chosen the Łukasiewicz logic because the semantics of its operators is more suitable for our framework. For these reasons we have excluded other well-known logics as Product Logic, in which for instance the negation operator is not involutive.

In this paper a new framework for a general theory of multivalued planning based on the infinite-valued Łukasiewicz logic [5, 9] is introduced and an algorithm for a multivalued planner is described. The proposed model can be seen as a generalization of the classical planning, since boolean and multivalued actions and fluents can be used in the same domain.

The paper is organized as follows. Some basic concepts on Łukasiewicz logic are recalled in Section 2, the model of multivalued planning is presented in Section 3 and an algorithm solving problems in such a model is proposed in Section 4. Finally an example, related works and conclusions are pointed out in Sections 5 and 6.

2 A brief introduction to Łukasiewicz logic

Łukasiewicz logic is a multivalued extension of classical logic, well known from the theoretical point of view [9, 5]. In this section we will report only some useful definitions. The conjunction operator, $x \odot y = \max(0, x + y - 1)$, has the properties of a generic *T*-norm: it is commutative, associative, monotonic with respect to all the variables; it has 0 as absorbing element and 1 as unit element.

¹ Università degli Studi di Perugia, Italy, email: {baioletti,milani,poggioni,suriani}@dipmat.unipg.it

The implication is defined as the residual of the conjunction operator and is expressed by $x \rightarrow y = \min(1, 1 - x + y)$.

The negation, which can be defined as $\neg x = x \rightarrow 0$, is expressed as $\neg x = 1 - x$ and is involutive. Using the last property and De Morgan's law it is possible to define a disjunction operator, expressed as $x \oplus y = \min(1, x + y)$. This operation is a co-norm: it is commutative, associative, monotonic monotonic with respect to all the variables; it has 1 as absorbing element and 0 as unit element.

Implication and disjunction are related by the condition $x \rightarrow y = \neg x \oplus y$.

It is possible to define an operation similar to subtraction, as $x \ominus y = x \odot \neg y = \max(0, x - y)$. Under some condition on x and y , it is the inverse of additive disjunction. Moreover $(x \ominus y) \odot z = x \ominus (y \oplus z)$ holds.

Lastly we introduce the lattices operators *min*-conjunction $x \wedge y = x \odot (x \rightarrow y)$ and *max*-disjunction $x \vee y = (x \rightarrow y) \rightarrow y$. They coincide, respectively, with the minimum and the maximum between x and y .

3 The Model

In this section a model of multivalued fluents and graded actions is presented. This model allows to use boolean fluents and classical actions jointly with these new kinds of fluents and actions.

3.1 States

Let \mathcal{F} be the finite set of fluents. A state is described by a function S which assigns to each fluent f a truth value belonging to $[0, 1]$ and denoted by $S(f)$.

It can be seen as a natural extension of the classical state definition in which the range of the state function is $[0, 1]$ instead of $\{0, 1\}$.

The value of a negated fluent is denoted by $S(\neg f)$ and it is computed as $S(\neg f) = \neg S(f)$. In general, given a generic logic formula ϕ , $S(\phi)$ denotes the truth value of ϕ in the state S .

A boolean fluent is a fluent whose possible values are restricted in $\{0, 1\}$.

3.2 Actions

Following the classical action definition introduced in STRIPS, an action a is described by a list of preconditions $pre(a)$ and a list of effects $eff(a)$.

Differently from the classical models, an application degree $\alpha(a) \in [0, 1]$ is assigned to each action a . The application degree of a boolean action is only 0 or 1, while for a multivalued action, $\alpha(a)$ can assume any intermediate value.

The degrees 0 and 1 have the same meaning as in the classical model (respectively, the action is not executed and the action is fully executed), while an intermediate value means that the action is executed with a lesser strength.

Informally we can say that each precondition is provided with a threshold. The action is fully executable only when the truth values of each precondition exceed the corresponding thresholds, while in the other cases the action can be executed with a smaller degree.

The execution of an action a can modify the truth value of fluents by means of additive changes: p is a positive effect of a if, after its execution, the value of p is increased, while p is a negative effect if the value is decreased. The amount of increment is proportional to the application degree of the action and can be tuned by a parameter which models the strength of the effect.

3.2.1 Action Preconditions and Action Executability

Definition 1 Action Preconditions

The preconditions of an action a are defined by a list of pairs (β_i, p_i) , where p_i is a literal (i.e. a fluent or its negation) and β_i is a real number in $[0, 1]$, called the threshold of p_i .

The threshold can be read as a coefficient of importance of p_i for a or a sort of a hardness degree of the precondition. $\beta_i = 0$ means that the precondition is *totally soft*, i.e. the truth value of the fluent does not affect the executability of a . While $\beta_i = 1$ means that the precondition is *totally hard*, i.e. that the truth value of p_i is required to be 1 for a to be fully executed.

If p_i is a boolean fluent, then β_i must be 0 or 1.

Definition 2 Action Executability

An action a is executable in a state S with application degree $\alpha \in [0, 1]$ if $\alpha \leq \alpha_{max}(a, S)$, where

$$\alpha_{max}(a, S) = \bigwedge_{(\beta, p) \in pre(a)} (\beta \rightarrow S(p)). \quad (1)$$

It is reasonable to compute a maximum application degree because the minimum application degree must be always 0, i.e. the planner can choose whether to apply a and, in this case, it can apply a with any actual application degree smaller or equal to $\alpha_{max}(a, S)$.

The *min*-conjunction \wedge is used in (1), instead of the usual Łukasiewicz conjunction \odot , because the latter, differently from the former, is not idempotent and if x and y are smaller than 1, $x \odot y$ is smaller than x and y . This property would cause some problems when there are some preconditions which are strictly related, modelling, for instance, the same feature of the world. In that case it is reasonable to expect that the presence of both the preconditions would not alter the executability of the action with respect to the case in which only one precondition is present.

The expression $\beta_i \rightarrow S(p_i)$ can be seen as a fuzzy extension of the crisp relation $S(p_i) \geq \beta_i$. In fact, if $S(p_i) \geq \beta_i$, then the implication assumes truth value 1, while if $S(p_i) < \beta_i$ then the implication assumes a value lesser than 1. The higher the difference between $S(p_i)$ and β_i is, the lesser the value of the implication is. The use of \rightarrow therefore allows to execute a (with $\alpha(a) < 1$) even if $S(p_i) \leq \beta_i$. Moreover, as desirable, this expression is increasing with respect to $S(p_i)$ and decreasing with respect to β_i .

Finally, it is important to note that a negative precondition has a particular semantics, i.e. it requires that the truth value of the fluent is small. In fact $\beta \rightarrow S(\neg p) = 1$ if and only if $S(\neg p) \geq \beta$, i.e. $S(p) \leq 1 - \beta$.

3.2.2 Action Effects and Action Execution

Definition 3 Action Effects.

The effects of an action a are defined by a list of pair (γ_i, e_i) , where e_i is a literal and γ_i is a real number in $[0, +\infty)$, called weight of a over e_i .

It is not allowed to have an action having both a positive and a negative effect over the same fluent.

Definition 4 Action Execution.

In the state S' , resulting from the execution of an action a with application degree α in a state S , for each effect (γ, e) of a

$$S'(e) = S(e) \oplus \Pi_\gamma(\alpha) \quad (2)$$

where $\Pi_\gamma(\alpha) = \min(\gamma \cdot \alpha, 1)$.

Since the aim of our model is to deal with actions which modify the world state according to their application degrees, additive effects are more suitable. For instance, if an action is executed with a low application degree, the change on the world has to be small. An action which assigns truth values to fluents depending only on α does not verify this important property.

The unary operator Π_γ allows to reduce or to amplify the effect of the action execution with respect to the application degree and, in particular, reflects the idea of effects proportionally related to the application degree.

It is easy to see that the application of (2) to a negated fluent $\neg f$ yields to $S'(\neg f) = S(\neg f) \oplus \Pi_\gamma(\alpha)$ and, since $S'(\neg f) = 1 - S'(f)$, to

$$S'(f) = S(f) \ominus \Pi_\gamma(\alpha).$$

Therefore, as expected, the action execution decrements the truth value of fluents in its negative effects, while increments the fluents in its positive effects.

If e_i is a boolean literal, then the corresponding γ_i is represented by ∞ because by this value it is possible to simulate assignments to 1 (or to 0).

3.2.3 Multivalued Planning Problems

A *multivalued planning problem* is defined by a quadruple $(I, \mathcal{O}, G, \sigma)$ where I is a truth assignment over the fluents $I : \mathcal{F} \rightarrow [0, 1]$ and denotes the initial state; \mathcal{O} is a set of boolean and multivalued actions; G is defined by a list of pairs (θ_i, g_i) having the same structure of action preconditions, and σ is a real number in $[0, 1]$ called *global threshold*.

A valid plan is an ordered sequence of pairs $\langle a_1 : \alpha_1, a_2 : \alpha_2, \dots, a_n : \alpha_n \rangle$, where each α_i represents the application degree of action $a_i \in \mathcal{O}$. Moreover, let $S_0 = I$ and S_{i+1} the state resulting from the execution of the action a_i is S_i , each action a_i must be executable in S_i with application degree α_i .

It is clear that non interfering actions could be simultaneously executed. While a boolean concept of interference among actions is natural, it is not straightforward to define a multivalued version of this concept and this point needs further investigations. Therefore in this first model we restrict our attention only to linear plans.

Definition 5 Solution Plan

A solution plan is a valid plan where in the final state S_n

$$\bigwedge_{(\theta, g) \in G} (\theta \rightarrow S_n(g)) \geq \sigma \quad (3)$$

Note that a solution plan is a plan which achieves all the goals in a multivalued fashion. First, the θ s represent the importance or hardness of each goal, similarly to what the β s represent for the action preconditions. Then, σ specifies a global parameter which represents a minimal satisfaction degree desired for the solution plan.

Since we are solving a planning problem by means of a MIP solver, it is straightforward to allow the definition of a metric function in terms of a linear function of the application degrees of some actions. Then a solution plan must also minimize the metric function.

4 An Algorithm for Multivalued Planning

The idea of the algorithm is to solve bounded multivalued planning problems described in the model presented in Section 3 in three steps.

At first, a planning graph [3] is constructed until either the upper bound U for time-steps is exceeded or a state verifying the necessary condition for the solution existence is reached. Then the graph representation is translated into a MIP problem (Mixed Integer Programming), and finally a MIP problem solver is called. If the MIP problem has a solution, then also the planning problem has a solution which is directly derived from the MIP problem solution. If the MIP problem has no solution, the main loop continues by adding a new level until a solution is found or the bound U is exceeded. It is easy to prove that this algorithm is correct and complete in the class of the U -bounded multivalued planning problems.

4.1 Planning Graph Construction

The planning graph built in our system is similar to the planning graph used in Graphplan and other planning systems. The planning graph is a $2T + 1$ -leveled graph $G = (V, E)$ whose vertex set is $V = F_0 \cup A_0 \cup F_1 \cup A_1 \cup \dots \cup F_T$, where F_t is the set of *multivalued fluents* at time $t = 0, \dots, T$ and A_t is the set of *graded actions* at time $t = 0, \dots, T - 1$. Each edge in E links either a fluent $f \in F_t$ to an action $a \in A_t$, such that $(\beta, f) \in \text{pre}(a)$, or an action $a \in A_t$ to a fluent $f \in F_{t+1}$, such that $(\gamma, f) \in \text{eff}(a)$.

Note that there are no edges denoting mutual exclusion relationship, since such a concept is not used in our framework.

In the following, $S_t(f)$ means the truth value of the fluent $f \in F_t$ at the time-step t and $\alpha_t^*(a)$ denotes the upper bound for the application degree α of the action $a \in A_t$.

The values of $S_0(f)$ and $\alpha_0^*(a)$ can be computed in an exact way, because the initial state is completely known and $\alpha_0^*(a)$, for each $a \in A_0$, can be computed directly using the formula 1.

When $t > 0$, since it is not known which action will be executed in each time-step, the truth value $S_t(f)$ for each fluent $f \in F_t$ is unknown and, for the same reason, it is not possible to compute a realistic value of $\alpha_t^*(a)$ for each action $a \in A_t$.

Anyway, it is possible to compute a lower bound $\underline{S}_t(f)$ and an upper bound $\overline{S}_t(f)$ for the value $S_t(f)$ by taking into account the maximum change (in the positive and in the negative cases) caused by any possible action executed at time $t - 1$.

In order to compute a tighter upper bound for $\alpha_t(a)$ which takes into account the fact that in each time-step only one action is executed, we must compute lower and upper bounds for $S_t(f)$ conditioned to the execution at time $t - 1$ of a given action.

Therefore for each fluent $f \in F_t$ and each action a , we define the interval $[\underline{S}_t^a(f), \overline{S}_t^a(f)]$ which provides a bound for $S_t(f)$ after the application of the action a at time $t - 1$. Clearly,

$$[\underline{S}_t(f), \overline{S}_t(f)] = [\min_{a \in A_{t-1}} \underline{S}_t^a(f), \max_{a \in A_{t-1}} \overline{S}_t^a(f)] \quad (4)$$

holds, where

$$\underline{S}_t^a(f) = \begin{cases} \underline{S}_{t-1}(f) & \text{if } (\gamma, \neg f) \notin \text{eff}(a) \\ \underline{S}_{t-1}(f) \ominus \Pi_\gamma(\alpha_{t-1}^*(a)) & \text{if } (\gamma, \neg f) \in \text{eff}(a) \end{cases}$$

$$\overline{S}_t^a(f) = \begin{cases} \overline{S}_{t-1}(f) & \text{if } (\gamma, f) \notin \text{eff}(a) \\ \overline{S}_{t-1}(f) \oplus \Pi_\gamma(\alpha_{t-1}^*(a)) & \text{if } (\gamma, f) \in \text{eff}(a) \end{cases}$$

Then, we define $\alpha_{bt}^*(a)$ as the maximum execution degree of the action a , computed considering $[\underline{S}_t^b(p), \overline{S}_t^b(p)]$ as the interval bounding the truth values at time t for each fluent p involved in the preconditions of the action a . In other words, $\alpha_{bt}^*(a)$ is the maximum

execution degree of the action a at the time t taking into account that the action b was executed at time $t - 1$.

$$\alpha_{bt}^*(a) = \bigwedge_{(\beta, p) \in pre(a)} (\beta \rightarrow \bar{S}_t^b(p)) \wedge \bigwedge_{(\beta, \neg p) \in pre(a)} (\beta \rightarrow \neg \underline{S}_t^b(p)).$$

Finally, we compute

$$\alpha_t^*(a) = \max_{b \in A_{t-1}} \alpha_{bt}^*(a). \quad (5)$$

It is easy to prove that the computation of $\alpha_t^*(a)$ directly from $\underline{S}_t(f)$ and $\bar{S}_t(f)$, would produce a higher estimate of the maximum application degree.

Note nevertheless that a complete implementation of the previous method is not feasible. In fact the computation of $\alpha_t^*(a)$ at a time t would require the computation of $\underline{S}_t^{a_0, \dots, a_{t-1}}(f)$ and $\bar{S}_t^{a_0, \dots, a_{t-1}}(f)$ for every possible sequence of actions a_0, \dots, a_{t-1} .

The first time-step where a solution can exist is the first one which verifies the goal condition

$$\bigwedge_{(\theta, f) \in G} (\theta \rightarrow \bar{S}_t(f)) \wedge \bigwedge_{(\theta, \neg f) \in G} (\theta \rightarrow \neg \underline{S}_t(f)) \geq \sigma. \quad (6)$$

4.2 Constraints Extraction

Since it has been shown, for instance in [8], that constraints expressed by operators of Łukasiewicz logic can be transformed in linear constraints on integer and real variables, the constraints contained in the planning graph built as explained in Section 4.1 can be compiled into a Mixed Integer Programming (MIP) problem.

The truth value of each fluent f and the application degree of each action a at time t are respectively denoted by the real variables x_{ft} and x_{at} which take values in $[0, 1]$. For boolean fluent and actions the range is restricted to $\{0, 1\}$.

Different kinds of constraint must be satisfied in order to have a solution plan according to the model presented in Section 3. They can be grouped in *action executability rules*, *action execution rules* and *goal satisfiability rules*. Moreover, according to the definition of a valid plan, we have to express a linearity condition for each plan called *plan linearity constraint*.

The number of constraints in the MIP problem will be a polynomial of the number of fluents and actions in the planning graph as shown in the following sections.

4.2.1 Action Executability Rules

At each time-step t , the actual application degree of an action a must be less or equal than the upper bound $\alpha_t^*(a)$. Therefore,

$$x_{at} \leq \min\left(\min_{(\beta, f) \in pre(a)} \{\beta \rightarrow x_{ft}\}, \min_{(\beta, \neg f) \in pre(a)} \{\beta \rightarrow \neg x_{ft}\}\right).$$

Therefore for each $a \in A_t$ and for every $t = 0, \dots, T - 1$

$$\begin{cases} x_{at} \leq (1 - \beta) + x_{ft} & \text{for all } (\beta, f) \in pre(a) \\ x_{at} \leq (1 - \beta) + (1 - x_{ft}) & \text{for all } (\beta, \neg f) \in pre(a) \end{cases}$$

The number of constraints added to MIP problem for each time-step is equal to the total number of preconditions of every action.

4.2.2 Action Execution Rules

The value of of a fluent f at a time-step $t + 1$, $x_{f,t+1}$, can be computed by the following equation which depends on the value of f at the previous time-step and on the execution of some action.

$$x_{f,t+1} = x_{ft} \oplus \left(\bigoplus_{(\gamma, f) \in eff(a)} \Pi_\gamma(x_{at}) \right) \ominus \left(\bigoplus_{(\gamma, \neg f) \in eff(a)} \Pi_\gamma(x_{at}) \right). \quad (7)$$

Note that since only one action is executed at each time-step, in (7) at most one term among $\Pi_\gamma(x_{at})$ is greater than zero and affects the value of x_{ft} .

The translation of (7) in MIP constraints is performed by means of the techniques shown in [8] and in [1]. The overall number of constraints added to the MIP problem for each time-step is proportional to the number of fluents.

4.2.3 Goal Satisfiability Rule

This rule derives directly from the goal condition defined in (6) and requires that each subgoal (θ, f) must be fulfilled with a truth value greater or equal to the global threshold. It is expressed by the formula

$$\bigwedge_{(\theta, f) \in G} (\theta \rightarrow x_{ft}) \geq \sigma. \quad (8)$$

As shown in subsection 4.2.1, this constraint can be encoded as

$$\begin{cases} (1 - \theta) + x_{ft} \geq \sigma & \text{for all } (\theta, f) \in G \\ (1 - \theta) + (1 - x_{ft}) \geq \sigma & \text{for all } (\theta, \neg f) \in G \end{cases}$$

The number of constraints added to the MIP problem is equal to the number of problem goals.

4.2.4 Linearity Plan Rule

With this constraint we express the condition that only one action can be executed in a given time-step, i.e. at each time t , $\exists! a \in A_t : x_{at} \neq 0$.

This constraint can be represented directly by the MIP system

$$\begin{cases} x_{at} \leq \lambda_{at} & \text{for all } a \in A_t \\ \sum_{a \in A_t} \lambda_{at} = 1 & \\ \lambda_{at} \in \{0, 1\} & \text{for all } a \in A_t \end{cases} \quad (9)$$

It is easy to see that this condition is true if and only if at each time-step t there exists at most an action a , such that $x_{at} \neq 0$.

The number of constraints added to the MIP problem for each time-step is equal to the number of actions plus one.

4.3 Implementation

A prototype implementation of the previously described algorithm has been made in C++, using as a MIP solver the open-source library *lpsolve*, which is not very efficient, although the preliminary results obtained on small examples are encouraging.

We expect that further code optimizations and the use of more efficient linear programming libraries, like CPLEX, will produce good results from the computational point of view.

Another different approach to solve multivalued planning problems without metric is a compilation into a Linear Arithmetic Logic problem, handled by some solvers like MathSAT [6].

5 An Example

In order to model domains and problems with multivalued fluents and graded actions some obvious extensions to PDDL3 were added.

In this small example we present a domain having generic objects that can be wet and must to be painted. There are also some grippers that can be wet and hold objects. These domain properties are represented by the following fluents: (*obj_wet o*), (*painted o*), (*gr_wet g*), (*holding o g*) where *o* is an object and *g* is a gripper.

This domain has four actions. *dry* is a graded action that can dry a free gripper. *paint* is a graded action that can paint an object held by a gripper. *pickup* and *putdown* are boolean actions that respectively pick an object up from the table and put an object down to the table. The property to be on the table is represented by (*on_table o*). A gripper can hold only one object, can pick an object up only if it is free and the object is on the table. It can put an object down only if it holds it. A gripper can be wet and can be safely used only if it is not too much wet.

This example cannot realistically be modeled by classical planning domains, because qualitative properties like “be wet” or “be painted” cannot suitably be represented by boolean propositions: a gripper could be “a little wet” or “just damp”. Soft preconditions allow, for instance, to execute the action *pickup(gr,obj)* even if the gripper is not totally dry. Moreover, the classical planning model is not adequate for an action like *dry(gr)* because it would be applied with the same strength, both when the gripper is *very wet* and when it is *a little wet*. These are the action descriptions:

```
(:action paint
  :type graded
  :parameters (?obj - object ?gr - gripper)
  :precondition (and (1 (holding ?obj ?gr)))
  :effect (0.8 (painted ?obj)) )
(:action dry
  :type graded
  :parameters (?gr - gripper)
  :precondition (0.8 (gr.wet ?gr))
  :effect (0.9 not(gr.wet ?gr)) )
(:action pickup
  :type boolean
  :parameters (?obj - object ?gr - gripper)
  :precondition (and (1 (free ?gr))
    (1 (on_table ?obj))
    (0.7 (not(gr.wet ?gr)))) )
  :effect (and (infinity not(free ?gr)) (infinity (not(on_table ?obj)))
    (infinity (holding ?obj ?gr))) )
(:action putdown
  :type boolean
  :parameters (?obj - object ?gr - gripper)
  :precondition (1 (holding ?obj ?gr))
  :effect (and (infinity (free ?gr)) (infinity (on_table ?obj)))
    (infinity (not(holding ?obj ?gr))) ) )
```

An example of a problem is presented. We suppose to have the goal to paint the objects minimizing the *dry* application degrees in all the plan.

```
(define (problem example)
  (:domain slippery_gripper)
  (:objects gr1-gripper gr2-gripper obj1-object obj2-object)
  (:init (= 1 (free gr1)) (= 1 (free gr2))
    (= 1 (on_table obj1)) (= 1 (on_table obj2))
    (= 0.2 (gr.wet gr1)) (= 0.5 (gr.wet gr2))
    (= 0.2 (obj.wet obj1)) (= 0.7 (obj.wet obj2))
    (= 0.3 (painted obj1)) (= 0.6 (painted obj2)) )
  (:goal (and (0.8 (painted obj1)) (0.7 (painted obj2))) )
  (:threshold 0.9)
  (:metric minimize (is-executed dry)) )
```

6 Related Works and Conclusions

One of the mainly related planning models is the non-Boolean approach introduced in [11]. In this model *propositions* denoting fuzzy relations and *flexible operators* are allowed. The main differences with our approach is that they use finite-valued fluents and the mapping between preconditions and effects is obtained by a set of dis-

joint conditional clauses, where each clause determines the satisfaction degree of the effects associated with it. Despite of the fuzziness of the approach, the activation of a conditional clause is crisp, i.e. in flexible planning slight differences in the satisfaction degree of pre-conditions can activate completely different set of effects. Finally the concept of graded action is not present in [11], where operators are applied only at fixed application degree, because only the satisfaction degree of preconditions determines the effects of the action. In our approach, instead, the planner decides which application degree to assign to an action among the available values.

It is easy to see that some types of resources can be encoded in our model by appropriate multivalued properties values and additive effects. For instance the quantitative resource *fuel* in a tank, varying between 0 and 100 litres, can be modeled by a multivalued fluent *tank_full* in [0,1] which represents, in some sense, a fuzzy normalization of the resource *fuel*. Operators which produce/consume *fuel* would increment/decrement the truth value of fluent *tank_full*.

Recently PDDL [7] has been extended in order to allow *soft* constraints and goals preferences; the planning problem consists then in finding an admissible solution which maximizes the preference function. It is worth noticing that, although preferences are not considered in our approach, they could be encoded by introducing explicit multivalued *preference fluents* into the operators, and by specifying in the goals the desired preference level.

An interesting direction for future work is to study the effects of building the planning model on a finite-valued logic.

Different and more efficient approaches to solve multivalued planning problems need to be considered, such as graph based algorithms or using mixed boolean-real solvers.

Finally, from an experimental point of view, it is necessary to develop a set of benchmark problems for significant domains involving graded actions in order to test the performance of the implementation.

REFERENCES

- [1] E. Balas, ‘Disjunctive programming’, *Annals of Discrete Mathematics*, **5**, 3–51, (1979).
- [2] S. Bistarelli, U. Montanari, and F. Rossi, ‘Semiring-based constraint satisfaction and optimization’, *Journal of ACM*, **44**(2), 201–236, (1997).
- [3] A. Blum and M. Furst, ‘Fast planning through planning graph analysis.’, *Artificial Intelligence*, **90**(1–2), 279–298, (1997).
- [4] M. Chechik, B. Devereux, A. Gurfinkel, and S. Easterbrook, ‘Multi-valued symbolic model-checking’, *ACM Transactions on Software Engineering and Methodology*, **12**(4), 1–38, (2003).
- [5] R.L.O. Cignoli, I.M.L. D’Ottaviano, and D. Mundici, *Algebraic Foundations of many-valued Reasoning*, Kluwer Academic Publisher, Dordrecht, Boston, London, 2000.
- [6] Marco Bozzano et al., ‘The mathsat 3 system’, in *Proc. CADE-20*, (2005).
- [7] A. Gerevini and D. Long. Plan constraints and preferences in pddl3. Tech. Rep., Dep. of Electronics for Automation, University of Brescia, Italy, August 2005.
- [8] R. Hähnle, ‘Proof theory on many-valued logic – linear optimization – logic design: Connections and interactions.’, *Soft Computing*, **1**(3), 107–119, (1997).
- [9] P. Hajek, *Mathematics of Fuzzy Logic*, Kluwer Academic Publisher, Dordrecht, Boston, London, 1998.
- [10] C. Liu, A. Kuehlmann, and M. Moskewicz, ‘Cama: a multi-valued satisfiability solver’, in *Proc. of IEEE/ACM Int. Conf. on CAD*, pp. 326 – 333, (2003).
- [11] I. Miguel, P. Jarvis, and Q. Shen, ‘Efficient flexible planning via dynamic flexible constraint satisfaction.’, *Engineering Applications of Artificial Intelligence*, **14**(3), 301–327, (2001).

Strong Cyclic Planning under Partial Observability

Piergiorgio Bertoli¹, Alessandro Cimatti¹, Marco Pistore²

Abstract. Strong Cycling Planning aims at generating iterative plans that implement trial-and-error strategies, where loops are allowed only so far as there is a chance to reach the goal. In this paper, we tackle the problem of Strong Cyclic Planning under Partial Observability, making three main contributions. First, we provide a formal definition of the problem. We point out that several degrees of solution are possible and equally interesting, depending on the admissible delay between achieving the goal and detecting that it has been achieved. Second, we present a family of planning algorithms that tackle the different versions of the problem. Third, we implement the algorithms using efficient symbolic representation techniques, and experimentally compare their performances.

1 Introduction

Planning for nondeterministic domains has lately received considerable attention, for its ability to relax some basic limitations of classical planning [9, 2, 11, 4, 8, 5].

Dealing with nondeterminism is very difficult for the key reason that, in general, in such a setting, a plan is associated with multiple runs. In *strong* planning, a solution must *guarantee* that the goal is achieved in a finite number of steps *for all runs* associated with a plan. Often, however, strong plans may not exist. It is therefore reasonable to address the problem of finding a *strong cyclic solution*, i.e. a conditional plan that implements a trial-and-error strategy, either achieving the goal in a finite number of steps, or going through a possibly infinite but *good* loop, where there is still the possibility of reaching the goal.

The problem of strong cyclic planning has been addressed so far under the hypothesis of full observability, see [10, 5, 8]. In this paper, we tackle the problem of strong cyclic planning under the more realistic hypothesis of *partial observability*. We make three main contributions. First, we provide a formal definition of the problem of strong cyclic planning under partial observability. This turns out not to be straightforward, since different interpretations are possible, and equally interesting. In particular, partial observability results in uncertainty at execution time, which may prevent or delay the detection of goal achievement. We distinguish different degrees of solutions where a plan guarantees to (i) achieve the goal and detect achievement at the same time, (ii) achieve the goal and detect achievement after some delay, and (iii) achieve the goal without ever detecting achievement. As a second contribution, we provide a family of correct and complete algorithms for tackling the different classes of problems. The algorithms are based on forward search over an and-or graph, where the “or” branching corresponds to the choice of ac-

tions and the “and” branching is the effect of observations. Search nodes contain information on *belief states*, i.e. sets of indistinguishable states compatible with the perceived observations. While sharing a conceptual structure similar to that for strong planning under partial observability presented in [2], and while close in spirit to some work on heuristic cyclic planning [6], our approach departs considerably from both of them, due to the need of handling both loops and delayed detection. As a final contribution, we implement the algorithms based on efficient symbolic data structures, and we compare their performance. To the best of our knowledge, this implementation is the first to be able to tackle strong cyclic planning problems, also considering relaxed goal detection requirements.

The paper is structured as follows. First, we provide a formal framework for planning in partially observable, nondeterministic domains, and we define the different interpretations of strong cyclic planning problems under partial observability. Then, we present the algorithms, we outline their implementation and discuss their performance. We wrap up by discussing related and future work.

2 Planning in Nondeterministic Domains

A planning domain is a generic transition system, defined in terms of its *states* (of which some are marked as *initial*), of the *actions* it accepts, and of the possible *observations* that it can exhibit. A *transition function* describes how executing an action leads from one state to possibly many different states. Finally, an *observation function* defines what observations are associated to each state of the domain.

Definition 1 (planning domain) A nondeterministic planning domain with partial observability is a tuple $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$, where:

- \mathcal{S} is the set of states.
- \mathcal{A} is the set of actions.
- \mathcal{O} is the set of observations.
- $\mathcal{I} \subseteq \mathcal{S}$ is the set of initial states; we require $\mathcal{I} \neq \emptyset$.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$ is the transition function; it associates to each current state $s \in \mathcal{S}$ and to each action $a \in \mathcal{A}$ the set $\mathcal{T}(s, a) \subseteq \mathcal{S}$ of next states.
- $\mathcal{X} : \mathcal{S} \rightarrow 2^{\mathcal{O}}$ is the observation function; it associates to each state s the set of possible observations $\mathcal{X}(s) \subseteq \mathcal{O}$.

An action a is executable in state s if $\mathcal{T}(s, a) \neq \emptyset$. We require that in each state $s \in \mathcal{S}$ there is some executable action, and that some observation is associated to each state $s \in \mathcal{S}$ (i.e., $\mathcal{X}(s) \neq \emptyset$).

If $Q \subseteq \mathcal{S}$, we write $\mathcal{X}(Q)$ for $\bigcup_{s \in Q} \mathcal{X}(s)$; also, we write $\mathcal{X}^{-1}(o)$ for $\{s \in \mathcal{S} \mid o \in \mathcal{X}(s)\}$.

¹ ITC-IRST, via Sommarive 18, 38100, Povo, Italy, e-mail: [\[bertoli,cimatti\]@irst.itc.it](mailto:[bertoli,cimatti]@irst.itc.it)

² DIT - University of Trento, via Sommarive 14, 38100, Povo, Italy, e-mail: pistore@dit.unitn.it

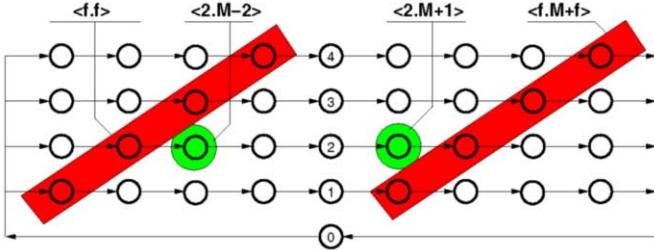


Figure 1. A simple explanatory robot domain

Example 2 Consider the domain composed of floors from 0 to M , and rooms from 0 to $R = 2 \times M$ (depicted in Fig. 1 for $M=4$). At the ground floor, composed of a single room, the robot can MoveUp with an elevator, which lands unpredictably at any floor. Then it can only MoveLeft, until it gets at room R , where it can MoveDown back to ground floor. All the rooms are indistinguishable, except for the middle room M of each floor, where a tag displays the current floor. The states of the domain are the pairs of the form $\langle f . r \rangle$, where $f \in 1 \dots M$ is the floor and $r \in 0 \dots R$ is the room of the robot, and $\langle 0 . M \rangle$ denotes the only location of the ground floor. The set of observations is $\{0, 1, \dots, M, \bullet\}$. The observation function is defined as follows: $\mathcal{X}(\langle f . r \rangle) = f$ iff $r = M$ (i.e., when in the room with the tag, the floor is perceived), and $\mathcal{X}(\langle f . r \rangle) = \bullet$ in the other locations (i.e. \bullet represents the absence of information).

We are interested in complex plans that may encode sequential, conditional and iterative behaviors. We model them as finite state machines, as follows.

Definition 3 (plan) A plan for planning domain $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$ is a tuple $\Pi = \langle \mathcal{C}, c_0, \alpha \rangle$, where:

- \mathcal{C} is the set of plan contexts.
- $c_0 \in \mathcal{C}$ is the initial context.
- $\alpha : \mathcal{C} \times \mathcal{O} \rightarrow 2^{\mathcal{A} \times \mathcal{C}}$ is the action relation; it associates to a plan context c and an observation o a set of pairs (a, c') , where a is an action to be executed and c' is a new plan context.

The *context* is the internal state of the plan. Actions to be executed depend on the context, and on the observation. Once an action is executed, the context is updated depending on the observation and on the action that has been carried out. Notice that our plans, in general, implement nondeterministic behaviors, i.e. different actions and new contexts may be taken (at different times during execution) when the plan is in the same context and receives the same observation. We call a plan deterministic when $\alpha(c, o)$ contains at most one pair (a, c') .

Example 4 With the plan $\Pi_{\text{ONE}}^{\langle f . M + f \rangle}$, the robot takes the elevator, moves to the middle room where it looks at the floor tag, and stops after f steps, where f is the number displayed on the tag. The plan has $R + 1$ contexts, of which S_{Init} is initial and T_0 is final; α is defined as follows:

c	o	a	c'	
S_{Init}	0	MoveUp	S_0	
S_i	\bullet	MoveLeft	S_{i+1}	$i = 0 \dots M - 1$
S_M	f	MoveLeft	T_{f-1}	
T_i	\bullet	MoveLeft	T_{i-1}	$i = 1..F$

The execution of a plan can be defined in terms of transitions between configurations that describe the state of the domain and of the plan, and it terminates when a context c is reached and an observation o is done for which $\alpha(c, o)$ is empty.

Definition 5 (configuration) A configuration for domain $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{T}, \mathcal{X} \rangle$ and plan $\Pi = \langle \mathcal{C}, c_0, \alpha \rangle$ is a tuple (s, o, c) such that $s \in \mathcal{S}$, $o \in \mathcal{X}(s)$, and $c \in \mathcal{C}$.

Configuration (s, o, c) may evolve into configuration (s', o', c') , written $(s, o, c) \rightarrow (s', o', c')$, if there is some action a such that:

- $(a, c') \in \alpha(c, o)$,
- $s' \in \mathcal{T}(s, a)$,
- $o' \in \mathcal{X}(s')$.

Configuration (s, o, c) is initial if $s \in \mathcal{I}$ and $c = c_0$. The reachable configurations for domain \mathcal{D} , plan Π , and goal \mathcal{G} , are defined as those in the transitive closure of \rightarrow from the initial configurations. Configuration (s, o, c) is terminal iff it is reachable and $\alpha(c, o) = \emptyset$.

Definition 6 (path) A finite [infinite] path P for a domain \mathcal{D} and a plan Π is a finite [infinite] sequence of configurations $\gamma_0, \gamma_1, \dots$ such that γ_0 is initial and $\gamma_i \rightarrow \gamma_{i+1}$ for all i . Path prefixes are defined in the usual way.

We denote with $\text{PATHS}(\mathcal{D}, \Pi)$ the set of the paths for \mathcal{D} and Π that either are infinite or end with a terminal configuration.

We are interested in plans that guarantee the executability of actions they produce on each reachable configuration.

Definition 7 (executable plan) Plan Π is executable on domain \mathcal{D} iff for each reachable configuration (s, o, c) , if $(a, c') \in \alpha(c, o)$, then $\mathcal{T}(s, a) \neq \emptyset$.

3 Defining Strong Cyclic Planning

We are interested in finding plans that achieve the goal despite initial state uncertainty and nondeterministic action effects. Strong planning [5] requires that this is achieved in a finite number of steps. Strong cyclic planning [5] relaxes this by admitting loops, so far as from each loop there still is a chance to reach the goal.

However, the definitions in [5] do not immediately lift to the case of partial observability. This is because while, in the fully observable case, after executing an action, we can always decide whether the goal has been reached, in the partially observable case, such “immediate” goal detection may be impossible.

In order to illustrate this point, we start by considering some problems in our example domain that can be solved by strong planning. If our goal is to reach and detect that the robot is at $\langle f . M + f \rangle$ (see Fig. 1, plan $\Pi_{\text{ONE}}^{\langle f . M + f \rangle}$) is a solution: the uncertainty about the position of the robot after the ride in the elevator disappears after looking at the sensors, so that at the end of every possible plan execution, we are sure that the desired condition is achieved. Consider now the goal of reaching one of the locations $\langle f . f \rangle$. With a plan $\Pi_{\text{ONE}}^{\langle f . M \rangle}$ that first takes the elevator and then takes M steps left, the robot will traverse the required state. However, the plan does not stop the robot in one of the required positions – in fact, this goal would be unsolvable. Thus, at the end of the plan execution, all we can guarantee is that “the goal has been achieved sometimes during the execution”. This example highlights the fundamental difference between a plan that only achieves a certain goal condition, and a plan that, in addition, guarantees that the executor will know when the goal is achieved (and will

stop accordingly). The first interpretation is the standard one used in strong planning under partial observability: a solution plan is associated with final belief states that are entirely contained in the goal. The second interpretation is new: a solution plan guarantees that, for each of the states in any final belief states, a goal state was previously visited. Accordingly, we distinguish between goal achievement *with immediate detection*, and goal achievement *with delayed detection*. Obviously, the requirement of immediate achievement detection is stronger than that of delayed achievement detection.

Let us now turn to the general case of strong cyclic planning. Consider the goal $\langle 2 \cdot M + 1 \rangle$. It is easy to see that it admits no strong solution in either interpretation. However, if we admit the possibility of infinite executions, the problem of goal achievement with immediate detection is solvable. A possible solution is given by a plan $\Pi_{\text{CYCLE}}^{(2 \cdot M + 1)}$ where the robot iterates taking the lift, traversing the rooms and descending, until it sees the floor 2 tag, and in that case it stops after moving left once. With such a plan, either the goal is reached (and immediately detected), or the system loops through states with a possibility of reaching the goal. Now consider the goal $\langle 2 \cdot M - 2 \rangle$. It is unsolvable under the immediate detection hypothesis. However there exists a strong cyclic solution with delayed achievement detection, e.g. a plan $\Pi_{\text{CYCLE}}^{(2 \cdot M)}$ where the robot iterates traversing taking the lift, traversing the rooms and descending, stopping as soon as it sees the floor 2 tag. Finally, consider a variation of the domain, where sensors are not precise and associate the same information to adjacent floors, and let our goal be $\langle 2 \cdot M + 1 \rangle$. Due to the robot being unable to localize its floor, it is not possible to achieve the goal and detect it (immediately or later). However a cyclic solution exists if goal detection is given up completely, by a plan Π_{CYCLE}^* that traverses the rooms without ever stopping.

We can formally characterize (strong and) strong cyclic solutions under partial observability, by first defining a notion of *observational equivalence* among paths which are indistinguishable under the point of view of an external observer:

Definition 8 (equivalent paths) Paths $P = (s_0, o_0, c_0), (s_1, o_1, c_1), \dots$ and $P' = (s'_0, o'_0, c'_0), (s'_1, o'_1, c'_1), \dots$ are said to be equivalent, denoted with $P \equiv P'$, iff for every i , $o_i = o'_i$.

Definition 9 (strong and strong cyclic solution) Let plan Π be executable in domain \mathcal{D} . Π is a strong solution to \mathcal{G} for

- achievement with immediate detection (SID) iff $\text{PATHS}(\mathcal{D}, \Pi)$ contains only finite paths, and $s \in \mathcal{G}$ holds for each terminal configuration (s, o, c) ;
- achievement with delayed detection (SDD) iff $\text{PATHS}(\mathcal{D}, \Pi)$ contains only finite paths, and each (finite) path in $\text{PATHS}(\mathcal{D}, \Pi)$ traverses some state $s \in \mathcal{G}$.

Π is a strong cyclic solution to \mathcal{G} for

- achievement with immediate detection (SCID) iff $s \in \mathcal{G}$ holds for each terminal configuration (s, o, c) , and each prefix of an infinite path in $\text{PATHS}(\mathcal{D}, \Pi)$ is observationally equivalent to some prefix of a finite path in $\text{PATHS}(\mathcal{D}, \Pi)$.
- achievement with delayed detection (SCDD) iff each finite path in $\text{PATHS}(\mathcal{D}, \Pi)$ traverses some state $s \in \mathcal{G}$, and each prefix of an infinite path in $\text{PATHS}(\mathcal{D}, \Pi)$ is observationally equivalent to some prefix of a finite path in $\text{PATHS}(\mathcal{D}, \Pi)$.
- achievement without detection (SCND) iff each (finite and infinite) path in $\text{PATHS}(\mathcal{D}, \Pi)$ traverses some state $s \in \mathcal{G}$.

```

1 function TopLevelPlanner(PBclass)
2   n.bs = I
3   n.os = I \ G
4   Gph.storeNode(n)
5   PBclass.propagateLabels(Gph)
6   while ( $\neg Gph.RootSolved() \wedge \neg Gph.EmptyFrontier()$ )
7     n := Gph.pickNode()
8     forall o  $\in \mathcal{O}.a \in A$ 
9       if  $((n.bs \cap \mathcal{X}^{-1}(o)) \neq \emptyset \wedge$ 
10         $\forall s \in (n.bs \cap \mathcal{X}^{-1}(o)) : \mathcal{T}(s, a) \neq \emptyset)$  then
11          let n'
12          n'.bs :=  $\mathcal{T}((n.bs \cap \mathcal{X}^{-1}(o)), a)$ 
13          n'.os :=  $\mathcal{T}((n.os \cap \mathcal{X}^{-1}(o)), a) \setminus G$ 
14          Gph.storeNode(n')
15          Gph.storeEdge((n, o, a, n'))
16        endif
17   PBclass.propagateLabels(Gph)
18   return (Gph.RootSolved())
19 end
```

Figure 2. The Generic Planning Routine

```

1 define ID.gotG(n) :=  $(n.bs \subseteq G)$ 
2 define ID.gotO(n) :=  $(n.bs \cap \mathcal{X}^{-1}(o) \subseteq G)$ 
3 define DD.gotG(n) :=  $(n.os = \emptyset)$ 
4 define DD.gotO(n) :=  $(n.os \cap \mathcal{X}^{-1}(o) = \emptyset)$ 
```

```

1 function SID.propagateLabels()
2   forall n  $\in Gph.nodes.$ 
3     label(n) := ID.gotG(n)
4   repeat on n  $\in Gph.nodes$ 
5     label(n) := label(n)  $\vee$ 
6      $(\forall o \in \mathcal{X}(n.bs) :$ 
7       ID.gotO(n)  $\vee \exists n' : Gph.edge(n, o, a, n') \wedge label(n'))$ 
8   until fixpoint for label(n)
9 end
```

Figure 3. Labeling Routine for SID

We remark that, even within a strong cyclic solution for SCND, there may exist paths for which (immediate/delayed) detection takes place. That is, there may be cases where an SCND solution terminates, just as well as for SCID and SCDD solutions; but as soon as detection cannot be always guaranteed, only SCND solutions may exist for the problem at hand.

4 Strong (Cyclic) Planning Algorithms

The algorithms for strong and strong cyclic planning share the same top-level structure, presented in Fig. 2. The TopLevelPlanner routine takes in input a problem class (SID, SDD, SCID, SCDD, SCND); \mathcal{D} and G are assumed to be globally available to the subroutines. The search iterates by extracting and expanding one node from the search graph and computing whether a solution exists; it returns when either all the nodes have been expanded (and the search frontier is empty), or when a solution has been found. Gph contains the search space explored so far. Intuitively, the search space is an and-or graph, where each edge represents a possible observation o (produced by the do-

main), and a subsequent action a (which can be executed given that observation). All through the search, each search node n is associated with two pieces of information: the belief state $n.bs \subseteq \mathcal{S}$ contains all the states that are compatible with the history of previous observations/actions, and the set of open states $n.os \subseteq n.bs$ stores the states of $n.bs$ that have not reached the goal in their past history.

The algorithms differ only in the labeling routine. In the strong cases, labeling is carried out by computing a least fix point (see Fig. 3 for the case of SID). The nodes are initially labeled depending on local information (line 3); then, for each node, the label is recomputed, based on the labels of connected nodes, until a stable condition is reached (lines 6-8). In the case of SID, a node is initially successful if its belief state is contained in \mathcal{G} (see $ID.gotG$), and a node labeled false becomes labeled true if, for every possible observation, either the belief state resulting after the observation is contained in \mathcal{G} (see $ID.gotO$), or there exists a successor node that is labeled true. In the case of SDD, it is initially successful if it has no open states, and it becomes successful if, for every possible observation, either the belief state resulting after the observation has no open states, or there exists a successor node that is labeled true. That is, the labeling routine for SDD can be obtained from the one for SID by substituting $ID.gotG$ and $ID.gotO$ with $DD.gotG$ and $DD.gotO$ respectively.

The main difficulty with the strong cyclic cases is in handling loops: it is not possible to consider a single fixed-point computation for labeling nodes as success. Admitting loops means that “weakly successful” nodes, from which termination is not guaranteed, must also be considered as taking part to possible solutions. Such nodes may be computed by a least fix point computation, starting from locally successful ones, and establishing that a node is “weakly successful” if it has at least one weakly successful successor. Once weakly successful nodes are computed, we have to discard loops where there is no chance to reach the goal. This can be performed by a subsequent greatest fix point computation, that sets as false the labels of nodes for which some observation does not lead to success (or to a successful successor). These two computations influence each other, and are interleaved until the situation stabilizes, i.e. a (higher-level) fix point is reached.

In the case of SCID (Fig. 4), the labeling function contains a high-level fix point computation at lines 4-20; within this, we first label nodes as weak solutions with the least fix point at lines 7-11; then, at lines 12-17, we relabel as not successful all those nodes for which success is not supported for every possible observation (either by a an action leading to a successful node, or by direct containment in the goal). The case of SCDD is analogous, and can be obtained by replacing the $ID.gotG$ and $ID.gotO$ success checks with $DD.gotG$ and $DD.gotO$ respectively.

The labeling routine for the case of SCND shares the same nested fix-point structure as the previous cases. However, it is more complicated, since it is no longer possible to detect whether a node can be labeled as success simply based on its belief state (as in SCID) or on its set of open states (as in SCDD). Thus, a node is no longer labeled with a truth value, but with a set of states (in particular, a subset of the open states). A state is in the label when it has been checked to satisfy the goal, i.e. all its possible future executions either reach \mathcal{G} or loop through states from which \mathcal{G} can be reached. A node is then successful iff its label is equal to its set of open states. The external fix point at lines 4-21 iterates by inserting into the label every weakly successful state (lines 7-12), and then (lines 13-18) by removing those states whose presence is not justified by the existence, for each observation, of an action that guarantees either reaching \mathcal{G} or a state (temporarily) labeled as success.

```

1 function SCID.propagateLabels()
2   forall  $n \in Gph.nodes$ 
3      $label(n) := true$ 
4   repeat
5     forall  $n \in Gph.nodes$ 
6        $newlabel(n) := ID.gotG(n)$ 
7     repeat on  $n \in Gph.nodes$ 
8        $newlabel(n) := label(n) \wedge$ 
9          $(\exists o \in \mathcal{X}(n.bs) : (ID.gotO(o)) \vee$ 
10         $\exists n' : Gph.edge(n, o, a, n') \wedge newlabel(n'))$ 
11   until fixpoint for  $newlabel(n)$ 
12   repeat on  $n \in Gph.nodes$ 
13      $newlabel(n) := newlabel(n) \wedge$ 
14        $\forall o \in \mathcal{X}(n.bs) :$ 
15          $(ID.gotO(o)) \vee$ 
16          $\exists n' : Gph.edge(n, o, a, n') \wedge newlabel(n'))$ 
17   until fixpoint for  $newlabel(n)$ 
18   forall  $n \in Gph.nodes$ 
19      $label(n) := newlabel(n)$ 
20   until fixpoint for  $label(n)$ 
21 end
```

Figure 4. Labeling Routine for SCID

```

1 function SCND.propagateLabels()
2   forall  $n \in Gph.nodes$ 
3      $label(n) := n.os$ 
4   repeat
5     forall  $n \in Gph.nodes$ 
6        $newlabel(n) := \emptyset$ 
7     repeat on  $n \in Gph.nodes$ 
8        $newlabel(n) :=$ 
9          $\{s \in label(n) \mid$ 
10         $\exists o \in \mathcal{X}(s). \exists n' : Gph.edge(n, o, a, n') \wedge$ 
11         $(\mathcal{T}(s, a) \cap (newlabel(n') \cup \mathcal{G}) \neq \emptyset)\}$ 
12   until fixpoint for  $newlabel(n)$ 
13   repeat on  $n \in Gph.nodes$ 
14      $newlabel(n) :=$ 
15        $\{s \in newlabel(n) \mid$ 
16          $\forall o \in \mathcal{X}(s). \exists n' : Gph.edge(n, o, a, n') \wedge$ 
17          $\mathcal{T}(s, a) \subseteq (newlabel(n') \cup \mathcal{G})\}$ 
18   until fixpoint for  $newlabel(n)$ 
19   forall  $n \in Gph.nodes$ 
20      $label(n) := newlabel(n)$ 
21   until fixpoint for  $label(n)$ 
22 end
```

Figure 5. Labeling Routine for SCND

All the reported algorithms are correct and complete; we omit the proofs due to lack of space. Notice that the algorithms, in general, produce nondeterministic plans. These can either be directly executed via a scheduler that must choose “fairly” amongst different possible actions, or transformed offline into deterministic plans. Producing nondeterministic plans is crucial to the effectiveness of the

algorithms, since the deterministic counterpart Π_D of a nondeterministic plan Π_{ND} may be exponentially larger than Π_{ND} , since the states of Π_D must encode the history of the traversed states of Π_{ND} .

5 Experimental Evaluation

We implemented all of the presented algorithms, leveraging on efficient BDD-based image primitives to progress sets of states (e.g. beliefs and open sets) by observations and actions, to intersect and check the entailment amongst sets of states, and to perform quantification operations on sets of states. To provide a common basis for comparing the algorithms, we did not introduce problem-specific optimizations, and we adopted a simple breadth-first heuristic to insure that plans are optimal and that the heuristic does not favor a given algorithm, cluttering our comparison.

In our tests, we considered the domain of the example, scaling up its size up to 100 floors (and 201 rooms), and a set of 6 different problems of increasing difficulty, so to test every combination of requirements on the strength of the plan and on the guarantee of detecting its success. In particular, we report here the results for the problem of reaching room $\langle \lceil M/2 \rceil . [3 \times M/2] \rangle$. For such problem, no strong solution exists; a strong cyclic solution with immediate detection exists, given by a plan along the lines of $\Pi_{CYCLE}^{\langle 2 \cdot M + 1 \rangle}$. The results are

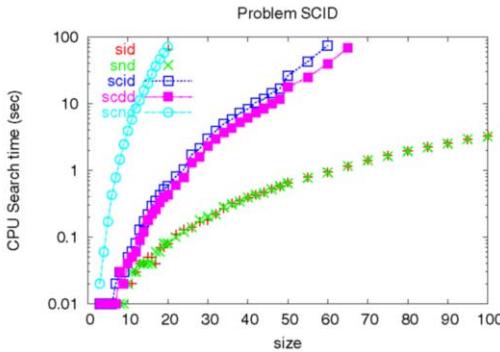


Figure 6. Results for the tests.

shown in Fig.6. A line-point style is used to indicate that the algorithm found a solution for the problem; only points are reported to indicate it signaled the absence of such a solution.

These results show clearly that strong algorithms perform much faster than strong cyclic ones. Moreover, amongst the strong cyclic algorithms, the one with no detection is far slower than those with immediate/delayed goal detection. This is mainly due to the different label propagation schema used by the algorithms: while the strong algorithms basically simply propagate boolean labels bottom-up on the graph, the strong cyclic algorithms with immediate/delayed detection compute such labels by interleaving an optimistic and a pessimistic fixed point computation, which may imply more computations of the label for a single node. For the SCND algorithm this becomes even far more complicated because the label is actually a set of states, and the number of interleaved fix-point computations may equal, in the worse case, the number of states in the beliefs. The remaining tests, which we do not report for reasons of space, consistently confirm the above intuitions.

6 Related works and Conclusions

In this paper we tackle the open problem of strong cyclic planning under partial observability. We provide a thorough, formal definition of the problem, and we present a family of algorithms which solve it under different requirements on the detection of goal achievement.

Amongst the many works in planning in nondeterministic domains [9, 2, 11, 4, 8], only few deal with strong cyclic planning ([5, 10, 8]), and, with the exception of [7], only for the much simpler case of full observability. The work in [7], on the other side, only provides an informal statement of the problem, and completely overlooks the issue of goal detection, constraining to the immediate detection case, and adopting a logical representation that sacrifices expressiveness to allow for a more effective implementation.

Our algorithms for strong cyclic planning share a nested least-greatest fix-point structure with LAO* [6], an algorithm for optimal search in and-or graphs. However, the extensions required to implement the termination checks cannot be easily encoded within the LAO* setting by means of probabilities, costs, and rewards. To solve strong cyclic planning under partial observability, LAO* would have to be changed by replacing its dynamic programming cost update procedure with a sub-procedure that tests whether the explicit and-or graph includes a valid cyclic plan. This is indeed what we devise, realizing our idea in a symbolic setting that allows for scalable algorithms by avoiding explicit state enumeration.

Strong cyclic planning could be thought of as a special case of the framework for planning under partial observability with extended goals presented in [1, 3]. This is not the case, since KCTL goals are not expressive enough to directly codify strong cyclic planning with delayed detection. Our algorithms, on the other hand, are amenable to the use of the efficient mechanisms for planning with nondeterministic domains (e.g. symbolic data structures, heuristics), and pave the way to an efficient implementation for planning for more general forms of temporally extended goals under partial observability.

REFERENCES

- [1] P. Bertoli, A. Cimatti, M. Pistore, and P. Traverso, ‘A Framework for Planning with Extended Goals and Partial Observability’, in *Proc. of ICAPS03*, (June 2003).
- [2] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso, ‘Planning in Non-deterministic Domains under Partial Observability via Symbolic Model Checking’, in *Proc. IJCAI’01*, (2001).
- [3] P. Bertoli and M. Pistore, ‘Planning with Extended Goals and Partial Observability’, in *Proc. of ICAPS04*, (June 2004).
- [4] B. Bonet and H. Geffner, ‘Planning with Incomplete Information as Heuristic Search in Belief Space’, in *Proc. of AIPS’00*, pp. 52–61. AAAI Press, (April 2000).
- [5] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, ‘Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking’, *Artificial Intelligence*, **147**(1-2), 35–84, (July 2003).
- [6] E. Hansen and S. Zilberman, ‘LAO*: a Heuristic Search Algorithm that finds Solutions with Loops’, *Artificial Intelligence*, **129**(1), 35–62, (2001).
- [7] L. Iocchi, D. Nardi, and M. Rosati, ‘Strong Cyclic Planning with Incomplete Information and Sensing’, in *Proc. of 4th International Workshop on Planning and Scheduling for Space*, (2004).
- [8] R. M. Jensen, M. M. Veloso, and M. H. Bowling, ‘OBDD-Based Optimistic and Strong Cyclic Adversarial Planning’, in *Proc. of ECP’01*, (2001).
- [9] F. Kabanza, M. Barbeau, and R. St-Denis, ‘Planning Control Rules for Reactive Agents’, *Artificial Intelligence*, **95**(1), 67–113, (1997).
- [10] Ugur Kuter, Dand Nau, Marco Pistore, and Paolo Traverso, ‘A Hierarchical Task Network Planner based Symbolic Model Checking’, in *Proc. of ICAPS-2005*, pp. 300–309, (2005).
- [11] J. Rintanen, ‘Constructing Conditional Plans by a Theorem-Prover’, *Journal of Artificial Intelligence Research*, **10**, 323–352, (1999).

Approximation Properties of Planning Benchmarks

Malte Helmert and Robert Mattmüller and Gabi Röger¹

Abstract. For many classical planning domains, the computational complexity of non-optimal and optimal planning is known. However, little is known about the area in between the two extremes of finding *some* plan and finding *optimal* plans. In this contribution, we provide a complete classification of the propositional domains from the first four International Planning Competitions with respect to the approximation classes **PO**, **PTAS**, **APX**, **poly-APX**, and **NPO**.

1 INTRODUCTION

Considering the important role that benchmark domains such as **LOGISTICS** and **SATELLITE** play in evaluating the performance of classical planning algorithms, comparatively little is known about their computational properties. With the notable exception of the **BLOCKSWORLD** domain [4, 12, 13], the published results on the computational complexity of planning benchmarks are rather coarse-grained. In most cases, they are limited to the analysis of two decision problems (relative to a certain planning domain):

- *Plan existence*: Is a given planning task solvable?
- *Bounded plan existence*: Is a given planning task solvable using no more than a certain number of actions?

For many planning domains, it turns out that the former problem is solvable in polynomial time, while the latter is **NP**-complete [5, 6]. In practice, this means that finding *some* plan for such a planning task is easy, while finding an *optimal* plan is difficult. In such a situation, it is natural to ask just how close to optimality we can get without sacrificing polynomial run-time.

This question has been raised and (partially) answered for many classical optimization problems such as the traveling salesperson problem, set covering problems, satisfiability-type problems, and countless others; it is the topic of the area of *approximation algorithms* [1]. In this contribution, we investigate the propositional planning domains introduced in the first four International Planning Competitions [2, 7, 9, 10] from this perspective, providing a complete classification with respect to the standard approximation classes.

We start our investigation by providing some background on the theory of approximation algorithms and how it applies to classical planning domains. We then present our main results in two parts, first discussing planning domains for which good, but not arbitrarily good, approximation algorithms exist, then planning domains for which good approximation algorithms do not exist. Finally, we summarize and conclude.

2 APPROXIMATION ALGORITHMS

Due to limited space, we keep our discussion of the theory of approximation algorithms short, pointing to the literature for formal

detail [1]. An *optimization problem* is either a *maximization problem* or a *minimization problem*; we focus on the latter exclusively. A minimization problem is given by a set of *instances*, a set of *feasible solutions* for each instance (possibly empty), and a *measure function* associating a number with each feasible solution of each instance. A solution is called *optimal* for a given instance if it minimizes the measure function among all feasible solution for that instance. An *approximation algorithm* for a minimization problem is a polynomial-time algorithm that, given an instance of the problem, generates a feasible solution for that instance or detects that none exists. An approximation algorithm that only generates solutions with measure at most c times the optimal measure (for some $c \in \mathbb{R}$) is called *c-approximating*. If such an algorithm exists, the problem is called *c-approximable*.

For the purposes of this paper, *planning domains* are minimization problems; their instances are called *planning tasks*. Solutions to planning tasks are sequences of *actions* called *plans*. The only measure function we consider is *sequential plan length*, which associates each plan with the number of actions it contains. Other common measure functions include parallel plan length, or weighted sequential plan length where some actions are more costly than others.

Optimization problems are grouped into different *approximation classes* in the same way that decision problems are classified into decision complexity classes like **P** and **NP**. Translating the usual definitions to the terminology of planning domains and tasks, the most important approximation classes are the following ones:

- Domains where optimal plans can be generated in polynomial time (**PO**).
- Domains which are *c*-approximable for *all* $c > 1$ (**PTAS**).
- Domains which are *c*-approximable for *some* $c > 1$ (**APX**).
- Domains where plans of length polynomially bounded by the optimal plan length can be generated in polynomial time (**poly-APX**).
- Domains where optimal plans can be generated in polynomial time by a non-deterministic Turing Machine (**NPO**).

It is easy to see that **PO** ⊆ **PTAS** ⊆ **APX** ⊆ **poly-APX** ⊆ **NPO** and that all these classes are identical if **P** = **NP**. More interestingly, all these inclusions are strict if **P** ≠ **NP** [1].

Membership of a minimization problem (planning domain) in one of these approximation classes can be shown constructively by providing a suitable approximation algorithm. Non-membership can be shown by *approximation-preserving reductions* (AP-reductions) from problems that are known to be hard, similar to the way many-one reductions are employed in classical complexity theory. Due to space restrictions, we do not present reductions in full formal details, so that we do not need to formally introduce AP-reductions.²

¹ Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany.
 Email: {helmert,mattmuel,roeger}@informatik.uni-freiburg.de

² Note to reviewers: Full proofs are available from the authors upon request via the program chair (brewka@informatik.uni-leipzig.de).

3 APPROXIMATION AND PLANNING

There is little work in the literature on the approximation properties of planning domains, the main exception being Selman's paper which links approximability to *reactive planning* and shows that the BLOCKSWORLD domains is 2-approximable and thus in **APX**, but not in **PTAS** unless $P = NP$ [12]. However, the literature does contain classifications of these planning domains from a classical complexity theory point of view [5, 6], addressing the *plan existence* and *bounded plan existence* decision problems. We can benefit from this body of work in the following ways:

- A domain belongs to **NPO** if shortest plan lengths are polynomially bounded by task size. Otherwise, it is not in **NPO**, as it is not always possible to *write down* a plan in polynomial time.
- Domains with polynomial-time planning algorithms belong to **poly-APX**, and to **PO** if the algorithms generate optimal plans.
- Domains with **NP-hard** *plan existence* problems do not belong to **poly-APX** unless $P = NP$.
- Domains with **NP-hard** *bounded plan existence* problems do not belong to **PO** unless $P = NP$.

Applying these rules to the complexity results for the benchmark domains of the four International Planning Competitions [5, 6], there are ten planning domains for which the classification into the approximation classes introduced in the previous section is not implied by the decision complexity results, namely BLOCKSWORLD, DEPOT, DRIVERLOG, GRID, LOGISTICS, MICONIC-SIMPLEADL, MICONIC-STRIPS, ROVERS, SATELLITE, and ZENOTRAVEL. In all these domains, polynomial non-optimal planning algorithms exist, but bounded plan existence is an **NP**-complete problem. Therefore, all these domains belong to **poly-APX**, but none belongs to **PO** unless $P = NP$. The open question, then, is whether these domains belong to **PTAS**, to **APX \ PTAS**, or to **poly-APX \ APX**.

As noted above, the approximation properties of BLOCKSWORLD have already been studied: it belongs to **APX**, but not to **PTAS** unless $P = NP$. In the following two sections, we present similar classification results for the remaining nine domains. We begin our investigation with domains which, like BLOCKSWORLD, are c -approximable but cannot be approximated arbitrarily well unless $P = NP$.

4 c -APPROXIMABLE DOMAINS

For most domains admitting polynomial planning algorithms, plans can be generated by simple greedy algorithms which satisfy the individual subgoals one after the other. If the number of steps required for each such individual goal can be bounded by a constant and an optimal plan requires at least one separate action per subgoal, then such an approach is c -approximating for some $c \in \mathbb{R}$. We first investigate a family of related domains sharing this property.³

Planning domain 1 SIMPLETRANSPORT

OBJECTS: Locations V , packages P , and vehicles T .

STATES: Each package has an associated location or is inside a vehicle. Each vehicle has an associated location. Initially, no package is inside a vehicle.

ACTIONS: Moving a vehicle between any two locations; picking up a package with a vehicle at the same location; unloading a package from a vehicle.

GOALS: Each package has an associated goal location.

The SIMPLETRANSPORT domain is not itself a planning competition benchmark, but it is closely related to some of them:

- **LOGISTICS** is a generalization of SIMPLETRANSPORT where vehicles are partitioned into *trucks* and *airplanes* and locations are partitioned into a set of *cities*, each with a distinguished *airport* location. Trucks may only move between locations in the same city, and airplanes may only move between (and be initially located at) airports.
- **MICONIC-STRIPS** is the SIMPLETRANSPORT domain restricted to a single vehicle. In the MICONIC domains, the vehicle is usually called an *elevator*, the locations *floors*, and the packages *passengers*. Instead of saying that a passenger is picked up or unloaded, we say that he *boards* or *leaves* the elevator.
- **MICONIC-SIMPLEADL** is identical to MICONIC-STRIPS except that pickup and unload actions are replaced by a single *stop* action. When this action is applied at location l , all passengers at l which are not at their goal location board the elevator, and all passengers inside the elevator with goal location l leave.
- **ZENOTRAVEL** is identical to SIMPLETRANSPORT except that each vehicle has an associated *fuel level* (a natural number). Vehicles can only move if their fuel level is non-zero, and movement reduces their fuel level by one.⁴ Refueling actions, increasing the fuel level of a vehicle by one, are applicable at any time.

It is quite easy to come up with c -approximation algorithms for these four domains. Indeed, for three of them, the greedy goal-at-a-time approach suffices. Therefore, all these domains belong to **APX**. We now show that it is hard to generate arbitrarily good plans.

Theorem 2 SIMPLETRANSPORT \notin **PTAS**, unless $P = NP$. This is true even in the special case with only one vehicle.

Proof sketch: Proof by AP-reduction from the MINIMUM VERTEX COVER problem for graphs where all vertices have degree 2 or 3, which is not in **PTAS** unless $P = NP$ [11]. A given graph $G = (V, E)$ is mapped to a SIMPLETRANSPORT task with one location for each vertex in V , plus one start location for the single vehicle. For each edge $\{u, v\} \in E$, one package must be moved from u to v and another one from v to u . The key observation is that the set of locations visited at least twice in a plan must form a vertex cover of G (i.e., includes at least one vertex from each edge in E). The degree restriction on the graphs guarantees that the size of a minimum vertex cover is $\Omega(|V|)$, which is important for proving that the reduction is approximation-preserving. ■

While the theorem shows that there must be *some* value $c > 1$ for which there is no c -approximating algorithm for SIMPLETRANSPORT with a single vehicle if $P \neq NP$, it does not provide us with an actual lower bound for approximability. In fact, the hardness result used for the reduction does not provide a lower bound either, but it is known that the MINIMUM VERTEX COVER problem for graphs with a vertex degree bound of 5 is not approximable within 1.0029 unless $P = NP$ [1]. By adjusting our reduction accordingly, we can exploit this result to show that MICONIC-STRIPS is not approximable within 1.000112 unless $P = NP$.

Corollary 3 LOGISTICS \in **APX \ PTAS**, unless $P = NP$.

MICONIC-STRIPS \in **APX \ PTAS**, unless $P = NP$.

MICONIC-SIMPLEADL \in **APX \ PTAS**, unless $P = NP$.

ZENOTRAVEL \in **APX \ PTAS**, unless $P = NP$.

³ To keep discussion short, we introduce planning domains informally and refer to the literature for exact PDDL definitions [2, 7, 9, 10].

⁴ There are also movement actions reducing fuel level by two, but there is no point in applying them.

Proof sketch: Greedy algorithms delivering one package (or passenger) at a time 2-approximate LOGISTICS and MICONIC-STRIPS and 3-approximate ZENOTRAVEL. (More sophisticated algorithms yield a $\frac{4}{3}$ -approximation for LOGISTICS and a $\frac{7}{6}$ -approximation for MICONIC-STRIPS.) These three domains are generalizations of SIMPLETRANSPORT with one vehicle, so they do not belong to PTAS unless $P = NP$.

MICONIC-SIMPLEADL is 2-approximable by an algorithm that moves to and stops at all initial floors of all passengers, then moves to and stops at all goal floors of all passengers. For proving non-membership in PTAS, essentially the same reduction as in the previous theorem can be used. ■

There are two more c -approximable benchmark domains. One of them is DEPOT, a combination of SIMPLETRANSPORT (packages are transported between locations by trucks) and BLOCKSWORLD (packages at each location are stacked into towers). The BLOCKSWORLD subproblem uses named and limited table positions; however, trucks provide unlimited auxiliary storage, so limitation of table positions is not problematic.⁵ Hardness for DEPOT follows from the BLOCKSWORLD and SIMPLETRANSPORT results immediately, and membership in APX is again fairly straightforward, so we do not define the domain formally but simply state our results briefly.

Theorem 4 $DEPOT \in APX \setminus PTAS$, unless $P = NP$.

Proof sketch: A straight-forward adaptation of the well-known 2-approximation for BLOCKSWORLD leads to a 3-approximation of DEPOT. The PTAS result follows because the domain generalizes both SIMPLETRANSPORT and BLOCKSWORLD. ■

Finally, we turn our attention to the SATELLITE domain, which is defined as follows.

Planning domain 5 SATELLITE

OBJECTS: Satellites S , instruments I , modes M , and directions D . Each instrument is located on one satellite, supports a set of modes, and has one direction as its calibration target.

STATES: Each satellite points at a certain direction. Each instrument may be powered on or off. Each instrument may or may not be calibrated. For each direction and mode, an image of this direction in this mode may or may not be available. Initially, no instrument is powered on or calibrated, and no images are available.

ACTIONS: Pointing a satellite at another direction; powering on an instrument (requires that no other instrument on this satellite is powered on; the instrument is not calibrated afterwards); powering off an instrument; calibrating an instrument (requires that its satellite points at its calibration target); taking an image of the pointing direction of a satellite in some mode (requires a powered-on, calibrated instrument supporting this mode on this satellite).

GOALS: Satellites may have associated target directions; images may need to be available for certain combinations of directions and modes.

SATELLITE falls into the same approximation category as the other domains in this section.

Theorem 6 $SATELLITE \in APX \setminus PTAS$, unless $P = NP$.

⁵ We require that there is at least one truck in each DEPOT task. DEPOT tasks without trucks would partition into sets of unrelated BLOCKSWORLD tasks with limited table positions. This BLOCKSWORLD variant is polynomially solvable [4], but we do not claim that it is in APX.

Proof sketch: A greedy algorithm solving one goal at a time (taking images first, then pointing satellites to their final directions, if any) is 6-approximating, showing membership in APX.

Limits of approximability can again be shown by an AP-reduction from MINIMUM VERTEX COVER for graphs with degree 2 or 3. A graph $G = (V, E)$ is mapped to a SATELLITE task with one satellite for each vertex $v \in V$ and one mode for each edge $e \in E$. The satellite corresponding to v has a single instrument supporting exactly the modes corresponding to edges incident to v . There is only one pointing direction d , which also serves as a calibration target for all satellites. An image of d must be taken in each mode. Given a plan for the task, the set of satellites used for taking images during the plan defines a vertex cover on G . The reduction is easily shown to be approximation-preserving. ■

Instead of using many satellites with a single instrument each, we could also prove hardness for the case of a single satellite with many instruments. Similarly, instead of requiring many images of one direction, we could also require one image each of many directions.

5 NON-APPROXIMABLE DOMAINS

In this section, we discuss the remaining three planning domains: ROVERS, DRIVERLOG, and GRID. The results we present are mostly negative: While it is easy to see that all these domains are in poly-APX, none of them admit constant-factor approximations unless $P = NP$. Like some of the domains we have seen previously, all three domains have a transportation or path-planning flavor. Differently to the domains we have seen previously, however, they do not exhibit a *fixed roadmap*. In the ROVERS domain, different vehicles use different roads. In DRIVERLOG, some edges can only be traversed by car, others only on foot. In GRID, locations can be initially inaccessible and need to be opened by using keys. We will see that these aspects of the problems are responsible for hardness of approximation, starting with the ROVERS domain.

Planning domain 7 SIMPLEROVERS

OBJECTS: Rovers R , waypoints W , and soil sample waypoints $S \subseteq W$. Each rover has an associated roadmap graph, which is a tree on a subset of waypoints. Each waypoint must be included in some roadmap graph. There is a waypoint common to all roadmap graphs called the lander location.

STATES: Each rover has an associated location, a waypoint on its roadmap graph. Rovers may carry a soil sample for a given waypoint or be empty. Soil data for a given waypoint may or may not have been transmitted. Initially, all rovers are empty and no soil data has been transmitted.

ACTIONS: Moving a rover between two waypoints connected on its roadmap; sampling soil on a soil sample waypoint with some rover (loads a soil sample into the rover, which must be empty; can only be done once for each waypoint); transmitting data for a soil sample with some rover (the soil sample must be inside the rover); emptying a rover.

GOALS: For each soil sample waypoint, a soil sample must be transmitted.

The domain is called SIMPLEROVERS because it excludes features of the full ROVERS domain such as other science targets besides soil samples (namely, rock samples and images), restricted visibility for transmitting science data, and rovers equipped for only a subset of science tasks [9]. Including these would require many more details without providing much further insight. Our results equally apply to the unrestricted domain.

Theorem 8 ROVERS $\in \text{poly-APX} \setminus \text{APX}$, unless $\mathbf{P} = \mathbf{NP}$. This is true even in the restricted SIMPLEROVERS domain.

Proof sketch: Any solvable ROVERS task can easily be solved one goal at a time, which shows membership in **poly-APX**. For the non-approximability result, we reduce from the MINIMUM SET COVER problem: Given a collection of subsets \mathcal{C} of a set S , find a sub-collection $\mathcal{C}' \subseteq \mathcal{C}$ with $\bigcup_{C' \in \mathcal{C}'} C' = S$, using the cardinality of the chosen sub-collection as the measure of the solution. The problem is not c -approximable for any $c \in \mathbb{R}$ unless $\mathbf{P} = \mathbf{NP}$ [1].

For a given instance (\mathcal{C}, S) , the corresponding SIMPLEROVERS task has one rover for each subset in \mathcal{C} , and one soil sample waypoint for each element of S . The roadmaps for the rovers are designed in such a way that the rover for $C \in \mathcal{C}$ can reach exactly those soil sample waypoints corresponding to elements of C . All soil sample waypoints reachable by a rover are very near to each other and a very long distance from the original rover location, so that plan lengths are dominated by the number of rovers used for solving the goals. Any such set of rovers defines a solution to the set covering instance; the reduction is approximation-preserving. ■

We presented this proof first because the other two are similar in spirit, but somewhat more involved technically. In both cases, the key idea is again that selecting a certain subset for the set cover allows achieving a set of goals corresponding to the elements of that subset, and the total length of the plan is dominated by the number of chosen subsets. This is enforced by requiring that a very large distance needs to be crossed in the planning task at least once for each chosen subset. We continue our exposition with the DRIVERLOG domain.

Planning domain 9 DRIVERLOG

OBJECTS: Trucks T , drivers D , major locations L , minor locations L_M (disjoint from L), and packages P . There is a connected road graph, whose vertices are the major locations, and a connected footpath graph, whose vertices are the major and minor locations. In the footpath graph, the degree of all minor locations is 2, and both adjacent vertices of a minor location are major locations.

STATES: A truck is located at some major location, a package at some major location or inside a truck, a driver at some major or minor location or inside a truck. Initially, all are located at major locations.

ACTIONS: Having a driver walk from one location to another (must be connected in the footpath graph); driving a truck from one location to another (must be connected in the road graph, must have a driver inside the truck); boarding a truck with a driver at the same location (must not have another driver inside the truck); debarking from a truck with a driver; loading a package into a truck at the same location; unloading a package from a truck.

GOALS: Packages, drivers, and trucks may have associated goal locations (which are always major locations).

The somewhat peculiar restriction on footpath graphs merely ensures that walking from one major location to another requires two actions rather than just one, to reflect the fact that walking usually takes longer than driving. There is never a good reason to walk from a major location to a minor location except to continue to the other adjacent major location, so instead of modeling minor locations explicitly, our following proof assumes that there is only one kind of location and assigns a cost of 2 to walk actions.

Theorem 10 DRIVERLOG $\in \text{poly-APX} \setminus \text{APX}$, unless $\mathbf{P} = \mathbf{NP}$. This is true even when there is only a single driver, and goals are only defined for packages.

Proof sketch: Again, a simple greedy algorithm solving one goal at a time suffices for showing membership in **poly-APX**. For hardness, we again provide an approximation-preserving reduction from MINIMUM SET COVER, mapping instance (\mathcal{C}, S) to the following DRIVERLOG task:

- There is a central location l_* , where the only driver starts. This is connected to *subset locations* l_C for each $C \in \mathcal{C}$ by very long disjoint footpaths. There is a truck at each subset location.
- For each element $s \in S$, there is an *element start location* l_s^- and an *element goal location* l_s^+ , and a road connecting the two. There is a truck and a package at each element start location. The package must be moved to the corresponding element goal location.
- For each subset $C \in \mathcal{C}$ and element $s \in C$, there is a location $l_{C,s}$ connected to l_C by a road and to l_s^- by a footpath.

It is easy to see that the subsets corresponding to the subset locations visited in a plan correspond to a set cover. By making the distances between central location and subset locations long enough, we can ensure that plan length is dominated by the cardinality of this set cover, so that the reduction is approximation-preserving. (The DRIVERLOG task does not have connected road and footpath graphs, but this can be repaired by introducing connecting paths that are too long to be useful for generating short plans.) ■

The last missing domain is GRID, another variation of the transportation theme.

Planning domain 11 GRID

OBJECTS: A single robot, locations L arranged into a rectangular grid, key shapes S , and keys K . Each key has a shape from S . A location may be initially locked by a lock with a certain shape from S . **STATES:** The robot has an associated location. Keys have an associated location or are carried. Locations can be open or locked. Initially, no key is carried.

ACTIONS: Moving the robot to an adjacent open location; picking up a key at the current robot location if no key is currently being carried; dropping a key being carried; switching the currently carried key with a key at the current robot location; opening a locked location adjacent to the robot with a key of the same shape as the lock. **GOALS:** Keys may have associated goal locations.

From our earlier analysis of the GRID domain, we know that GRID $\in \text{poly-APX}$ [5]. This reference also contains two proofs of hardness for the bounded plan existence problem in this domain, one based on satisfiability and one based on a traveling-salesperson type problem. However, neither of these reductions can be used to prove hardness of approximation. Indeed, the GRID tasks generated by the satisfiability reduction can easily be 2-approximated, and the restricted version of the GRID domain considered in the traveling salesperson reduction even belongs to **PTAS**. We thus require a new reduction.

Theorem 12 GRID $\in \text{poly-APX} \setminus \text{APX}$, unless $\mathbf{P} = \mathbf{NP}$.

Proof sketch: Membership in **poly-APX** is known [5]. Again, we map MINIMUM SET COVER instances (\mathcal{C}, S) to planning tasks via an approximation-preserving reduction. The GRID task has three kinds of keys: for each subset $C \in \mathcal{C}$, there is a *subset key* k_C with key shape S_C ; for each subset $C \in \mathcal{C}$ and element $s \in C$, there is an *element key* $k_{C,s}$ with key shape S_s ; for each element $s \in S$, there is a *goal key* k_s with a key shape that cannot be used for opening any location. Goals are only defined for the goal keys.

The grid of the planning tasks consists of two areas which are very distant from each other. The first area contains the initial locations of the element and goal keys and the robot as well as the goal locations of the goal keys. The second area contains the initial locations of the subset keys. The initial location of element key $k_{C,s}$ is locked with a lock of shape S_C , and the initial location of goal key k_s is locked with a lock of shape S_s . These are the only locations which are initially locked.

To solve the task, the robot must unlock the initial locations of all goal keys, which requires obtaining keys of shape S_s for all $s \in S$. This in turn requires obtaining subset keys for a collection of subsets that covers S , so the set of subset keys picked up in a plan defines a set cover. By putting a large enough distance between the first and second area, we can ensure that the cardinality of this set cover dominates total plan length. (Note that the robot can only carry one key at a time.) Thus, the reduction is approximation-preserving. ■

We point out that the hardness of approximately solving GRID tasks is largely due to the necessity of opening locations. If we restrict ourselves to tasks where opening is impossible (i.e., the shapes of locks are disjoint from the shapes of keys), we can provide a $(2 + \varepsilon)$ -approximating planning algorithm for all $\varepsilon > 0$. However, we will not prove this result here, but instead turn to discussion.

6 SUMMARY AND CONCLUSION

The approximation properties of the benchmark domains are summarized in Fig. 1. There are some interesting observations to be made. First, there are no domains in the benchmark set that fall into the class **PTAS** \ **PO**. There is no fundamental reason why this should be the case, except maybe for the fact that **PTAS** problems are rare in general.⁶ However, as we pointed out when discussing the GRID domain, there are **NP**-hard special cases of the competition domains that do admit polynomial-time approximation schemes.

Looking into the hardness proofs, we observe that we could easily get away with only using two different optimization problems for our reductions, both of which are set covering problems. We believe that this is no coincidence: Set covering problems arise naturally in planning tasks through positive interactions of subgoals. We believe that finding a good way of integrating heuristics for set covering problems into admissible heuristics for domain-independent planning could lead to a significant advance for optimal planning algorithms. However, we consider this a very challenging avenue of research.

What to do with these results? After our earlier results [5, 6] on the decision complexity of planning in the benchmark domains, we see our work as a second step on the road towards a better understanding of the benchmark suite, which we believe to be critical to provide “the level of understanding required for its effective use as a benchmark” [13, Slaney and Thiébaux on BLOCKSWORLD]. In comparison to Slaney and Thiébaux’s effort, at least two more steps are required until we can consider these standard benchmarks “understood”: one is the identification of phase transition regions to know where the hard instances are, and the other is the provision of (domain-dependent) optimal solvers as a reference point for evaluation of solution quality.

In addition to such deeper domain-specific studies, we believe that some important insights into domain-*independent* planning can be

⁶ To readers familiar with other approximation classes, we point out that there is a good reason why there is no domain in the class **FPTAS** \ **PO**; this follows quite easily from Theorem 3.15 in the book by Ausiello et al. [1]. Similarly, we cannot have planning domains in the class **exp-APX** \ **poly-APX**.

$\in \mathbf{PO}$:

GRIPPER, MOVIE, PROMELA-OPTICALTELEGRAPH,
PROMELA-PHILOSOPHERS, PSR, SCHEDULE

$\in \mathbf{PTAS} \setminus \mathbf{PO}$ (unless $\mathbf{P} = \mathbf{NP}$):

none

$\in \mathbf{APX} \setminus \mathbf{PTAS}$ (unless $\mathbf{P} = \mathbf{NP}$):

BLOCKSWORLD, LOGISTICS, MICONIC-SIMPLEADL,
MICONIC-STRIPS, ROVERS, SATELLITE, ZENOTRAVEL

$\in \mathbf{poly-APX} \setminus \mathbf{APX}$ (unless $\mathbf{P} = \mathbf{NP}$):

DEPOT, DRIVERLOG, GRID

$\in \mathbf{NPO} \setminus \mathbf{poly-APX}$ (unless $\mathbf{P} = \mathbf{NP}$):

FREECELL, MICONIC-FULLADL, MPRIME, MYSTERY,
PIPESWORLD*

$\notin \mathbf{NPO}$:

AIRPORT, ASSEMBLY

Figure 1. Classification results. For PIPESWORLD, only hardness is known; membership in **NPO** is open.

obtained by generalizing the analyses conducted for the benchmark domains in a suitable way, for example by identifying syntactic or semantic fragments of PDDL for which the planning problem belongs to **APX**, in the spirit of the work on tractable subclasses of the SAS⁺ planning formalism [3, 8].

REFERENCES

- [1] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi, *Complexity and Approximation*, Springer-Verlag, 1999.
- [2] Fahiem Bacchus, ‘The AIPS’00 planning competition’, *AI Magazine*, **22**(3), 47–56, (2001).
- [3] Christer Bäckström and Bernhard Nebel, ‘Complexity results for SAS⁺ planning’, *Computational Intelligence*, **11**(4), 625–655, (1995).
- [4] Narenesh Gupta and Dana S. Nau, ‘On the complexity of blocks-world planning’, *Artificial Intelligence*, **56**(2–3), 223–254, (1992).
- [5] Malte Helmert, ‘Complexity results for standard benchmark domains in planning’, *Artificial Intelligence*, **143**(2), 219–262, (2003).
- [6] Malte Helmert, ‘New complexity results for classical planning benchmarks’, in *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, eds., Derek Long and Stephen Smith. AAAI Press, (2006). To appear.
- [7] Jörg Hoffmann and Stefan Edelkamp, ‘The deterministic part of IPC-4: An overview’, *Journal of Artificial Intelligence Research*, **24**, 519–579, (2005).
- [8] Peter Jonsson and Christer Bäckström, ‘State-variable planning under structural restrictions: Algorithms and complexity’, *Artificial Intelligence*, **100**(1–2), 125–176, (1998).
- [9] Derek Long and Maria Fox, ‘The 3rd International Planning Competition: Results and analysis’, *Journal of Artificial Intelligence Research*, **20**, 1–59, (2003).
- [10] Drew McDermott, ‘The 1998 AI Planning Systems competition’, *AI Magazine*, **21**(2), 35–55, (2000).
- [11] Christos H. Papadimitriou and Mihalis Yannakakis, ‘Optimization, approximation, and complexity classes’, *Journal of Computer and System Sciences*, **43**, 425–440, (1991).
- [12] Bart Selman, ‘Near-optimal plans, tractability, and reactivity’, in *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR’94)*, eds., Jon Doyle, Erik Sandewall, and Pietro Torasso, 521–529, Morgan Kaufmann, (1994).
- [13] John Slaney and Sylvie Thiébaux, ‘Blocks World revisited’, *Artificial Intelligence*, **125**, 119–153, (2001).

Approximate linear-programming algorithms for graph-based Markov decision processes

Nicklas Forsell^{1,2} and Régis Sabbadin²

Abstract. In this article, we consider a form of compact representation of MDP based on graphs, and we propose an approximate solution algorithm derived from this representation. The approach we propose belongs to the family of *Approximate Linear Programming* methods, but the graph-structure we assume allows it to become particularly efficient. The proposed method complexity is linear in the number of variables in the graph and only exponential in the width of a dependency graph among variables.

1 INTRODUCTION

Markov Decision Processes (MDP) have become a standard model for decision-theoretic planning. However, classical dynamic programming algorithms [7] do not scale to the size of problems that can be compactly represented by the factored representations traditionally used in AI. Therefore several methods of aggregation/decomposition of large problems have been proposed in the AI community. Aggregation methods ([1, 4]) use a compact representation of the probability and reward functions involved in the MDP, and the proposed algorithms are designed to explicitly work with these factored descriptions.

In this article, we consider a form of compact representation of MDP based on graphs, and propose an approximate solution algorithm derived from this representation. The algorithm belongs to the family of *Approximate Linear Programming* methods and the graph-structure of the representation allows the algorithm to become particularly efficient. While previously proposed algorithms typically have a complexity exponential in the number of variables in the graph, our algorithm's is linear in the number of variables and only exponential in the width of a dependency graph among the variables.

First, we will describe the *Approximate Linear Programming* (ALP) method for solving large MDP [3]. We will then present the Graph-based multidimensional MDP framework (GMDP). The state and action variables of these MDP are multidimensional and attached to the nodes of a graph G allowing the representation of local dynamics. In this class of problems, it is assumed that the dynamics of each state variable only depend on the action applied locally and on the current values of the neighbouring state variables. We then introduce an extension of the ALP method, dedicated to solving Graph-based MDP. The methods complexity is linear in the number of variables in the GMDP, and exponential in the width of the dependency graph. To illustrate the method, we use a forest management example involving several treatment units, which will be referred to as stands.

¹ SLU, Umeå, Sweden, nicklas.forsell@resgeom.slu.se

² INRA-MIA, Toulouse, France, {forsell, sabbadin}@toulouse.inra.fr

2 MARKOV DECISION PROCESSES

2.1 Problem formulation

In its classical formulation [7], a stationary *Markov Decision Process* (MDP) is defined by a four-tuple $\langle \mathcal{X}, \mathcal{A}, p, r \rangle$, where \mathcal{X} represents the finite set of admissible states of the system, \mathcal{A} the finite set of applicable actions, $p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$ the transition probabilities between states, and $r : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ an “immediate” reward function. Note that both $p(x'|x, a)$ and $r(x, a)$ are independent of the decision stage t .

A function $\delta : \mathcal{X} \rightarrow \mathcal{A}$ assigning at each time step an action to every state, is called a (stationary) *policy*. The *value function* of a policy $v_\delta : \mathcal{X} \rightarrow \mathbb{R}$ is defined so that $v_\delta(x)$ represents the infinite horizon, discounted cumulative reward attached to a policy δ applied to a MDP, with initial state x . This value can be computed as:

$$v_\delta(x) = E \left[\sum_{t=0}^{\infty} \gamma^t r(x^t, \delta(x^t)) | x^0 = x \right]. \quad (1)$$

The expectation is taken over all possible trajectories $\tau = \langle x^0, a^0, x^1, \dots, x^t, a^t, \dots \rangle$, where from the initial state x , the policy δ is applied. The factor which ensures that the above infinite sum converges is the discount factor $0 \leq \gamma < 1$.

The problem of finding an optimal policy with respect to the discounted criterion (1) can be written as:

$$\text{Find } \delta^* : \mathcal{X} \rightarrow \mathcal{A}, \text{ so that: } v_{\delta^*}(x) \geq v_\delta(x), \forall x \in \mathcal{X}, \forall \delta \in \mathcal{A}^\mathcal{X}.$$

This problem is classically solved by Stochastic Dynamic Programming algorithms [7]. In the next section we present a method based on *linear programming* (see e.g. [7]).

2.2 Linear Programming Solution for MDP

In this paper, we will use the following definition of the *Bellman operator* \mathcal{B} operating on any value function v .

Definition 1 (Bellman operator) Let $v : \mathcal{X} \rightarrow \mathbb{R}$, the Bellman transform $\mathcal{B}(v) : \mathcal{X} \rightarrow \mathbb{R}$ is defined as:

$$\mathcal{B}(v)(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x, a)v(x') \right\}, \forall x \in \mathcal{X}.$$

Let v^* denote the value function attached to an optimal policy δ^* ($v^* = v_{\delta^*}$). It has been shown (see, e.g. [7]) that v^* can be computed as the unique solution of the system of non-linear equations $v^* = \mathcal{B}(v^*)$:

$$v^*(x) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x, a)v^*(x') \right\}, \forall x \in \mathcal{X}.$$

The optimal policy δ^* can then be obtained *greedily* from v^* :

$$\delta^*(x) = \arg \max_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x, a) v^*(x') \right\}, \forall x \in \mathcal{X}.$$

There exist several methods for computing v^* . We will focus our discussion on the *linear programming* approach. This method defines v^* as the solution of the following linear programming problem:

$$\begin{aligned} \min & \quad \sum_{x \in \mathcal{X}} c(x)v(x) \\ \text{s.t.} & \quad v(x) \geq r(x, a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x, a)v(x') \\ & \quad \forall x \in \mathcal{X}, \forall a \in \mathcal{A} \\ & \quad v(x) \text{ free, } \forall x \in \mathcal{X}. \end{aligned} \quad (2)$$

where $c(x), \forall x \in \mathcal{X}$ are arbitrary strictly positive numbers. This formulation of the problem contains $|\mathcal{X}|$ variables, $|\mathcal{X}| \times |\mathcal{A}|$ constraints and is commonly referred to as the *exact LP*. Note that any values for $c(x_1), \dots, c(x_{|\mathcal{X}|})$, will lead to the same optimal value function v^* .

2.3 Approximate Linear Programming (ALP)

For many practical problems, a state is described by the values of several state variables. Related to this gain in the power of expression, is the fact that the size of the state space grows exponentially with the number of state variables. In order to solve practical problems with more than a few variables (and thus of considerable size), using the Linear Programming method, the number of variables and constraints has to be decreased. The *approximate linear programming* (ALP) approach to MDP [3], looks for a *parameterised value function* v_w , which is an approximation of the optimal value function v^* . The parameterised value function v_w , is defined as a linear combination of a set of given basis functions $H = \{h_1, \dots, h_k\}$, $h_i : \mathcal{X} \rightarrow \mathbb{R}$, that can be written as:

$$v_w(x) = \sum_{i=1}^k w_i h_i(x), \forall x \in \mathcal{X}. \quad (3)$$

where $w = (w_1, \dots, w_k)$ is a vector of real-valued weights. Restricting the search of a solution of the LP (2), to the set of parameterised value functions (3), leads to a new *approximate LP model*:

$$\begin{aligned} \min & \quad \sum_{x \in \mathcal{X}} (c(x) \sum_{i=1}^k w_i h_i(x)) \\ \text{s.t.} & \quad \sum_{i=1}^k w_i h_i(x) \geq \\ & \quad r(x, a) + \gamma \sum_{x' \in \mathcal{X}} (p(x'|x, a) (\sum_{i=1}^k w_i h_i(x'))), \\ & \quad \forall x \in \mathcal{X}, \forall a \in \mathcal{A} \\ & \quad w_i \text{ free, } i = 1, \dots, k. \end{aligned} \quad (4)$$

The variables of the LP (4) are now the weights w_i . The problem has been transformed into the problem of finding a set of parameters values $w^* = (w_1^*, \dots, w_k^*)$, so that v_{w^*} be a close approximation of v^* . The advantage of this representation is that the number of variables has been decreased from $|\mathcal{X}|$ to k , the number of basis functions. Thus, the complexity of the problem has been reduced, to the price of the loss of the optimality of the obtained solution. Unfortunately the number of constraints is still $|\mathcal{X}| \times |\mathcal{A}|$, which means that a direct application of LP solution methods is still inefficient for solving really large problems. Several algorithms have been suggested for dealing with the exponential number of constraints induced by the ALP formulation of large MDP. For example [3], use the fact that most of the constraints in (4) are not active. They study an approximation of (4) in which a randomly chosen subset of constraints of tractable size is considered. They also provide probabilistic bounds

on the quality of the returned solution. [4], instead, use a factored representation of the constraints of the LP and a form of *variable elimination* for solving this factored LP.

In the following sections we will describe a specific class of problems for which we apply ALP techniques. Considering this restricted class, we will avoid the problem of facing exponential sets of constraints by *decomposing* the ALP model. The factorisation properties of the class of problems we study allow a further approximation of the ALP model. This in turn allows a replacement of the unique ALP model with a set of *independent* ALP models, of reasonable size.

3 GRAPH-BASED MARKOV DECISION PROCESSES

Factored MDP offer a compact representation of MDP with multidimensional state space, structured transition probabilities and structured reward functions. For solving factored MDP, a number of efficient algorithms relying on the structure of the problem, have been proposed. They are variants of either *value iteration* or *policy iteration* [1] or of ALP approaches [3, 4]. For example [4], propose an ALP-based algorithm relying on the structure of factored MDP in order to solve (4). However, one should note that in general the action spaces are supposed to be “flat” and the possible structure of the action space is not used in the solution algorithm. In this section we present a specific class of factored MDP, called *Graph-based Markov Decision Processes*. Even though it is a subclass of factored MDP, the multidimensionality of the action space is explicitly described and will be exploited by the ALP type of solution algorithms which will be described in the next section.

3.1 Definitions

The *Graph-based Markov Decision Process* (GMDP) is defined by a tuple $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$. The state and action spaces are now Cartesian products $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$, where \mathcal{X} and \mathcal{A} have the same dimensionality. The reason for \mathcal{X} and \mathcal{A} to have the same dimensionality, is for the transition and rewards to be defined as local (see definitions below). The *directed graph* $G = (V, E)$ expresses *local* dependencies among state and action variables.

A *neighbourhood function* N over V is defined as:

Definition 2 (Neighbourhood function) $N : V \rightarrow 2^V$ is defined as: $N(i) = \{j \in V | (j, i) \in E\}, \forall i \in V$.

Transitions and rewards are said to be *local* iff:

Definition 3 (Local transitions)

$$p(x'|x, a) = \prod_{i=1}^n p_i(x'_i | x_{N(i)}, a_i), \forall x' \in \mathcal{X}, x \in \mathcal{X}, a \in \mathcal{A}$$

where: $x_I = \{x_j | j \in I\}, \forall I \subseteq \{1, \dots, n\}$.

Definition 4 (Local rewards)

$$r(x, a) = \sum_{i=1}^n r_i(x_{N(i)}, a_i), \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$$

In the general case, policies for a GMDP take the form $\delta = (\delta_1, \dots, \delta_n)$, where $\delta_i : \mathcal{X} \rightarrow \mathcal{A}_i$. Nevertheless, global policies can take space in $O(n\alpha^{\sigma^n})$ where $\alpha = \max_i |\mathcal{A}_i|$ and $\sigma = \max_i |\mathcal{X}_i|$.

Except for problems of very small dimension, this clearly prohibits the computation of global policies. Some special policies, called *local policies* are therefore of interest:

Definition 5 (Local policy) A policy $\delta : \mathcal{X} \rightarrow \mathcal{A}$, is said to be local iff $\delta = (\delta_1, \dots, \delta_n)$, where $\delta_i : \mathcal{X}_{N(i)} \rightarrow \mathcal{A}_i$.

Even though the optimal policy of a GMDP may not be local³, it is interesting to look for "good" local policies. This since they are both easy to express (space complexity in $O(n\alpha^\sigma)$) and implement. We have also observed that in practice they can perform very well. Furthermore, as will be shown in Section 4, it is possible to find an efficient algorithm, based on ALP, for finding efficient local policies.

3.2 Example GMDP

In some areas of the world, damage to forests due to wind is considered a major problem. This since recent and earlier gales have wind-felled massive amounts of forests. For example, a storm in 2005, wind-fell approximately 75 millions cubic meter of wood in the southern parts of Sweden. This corresponds to the total normal harvest in Sweden per annum. Forest management activities performed in the stands composing the forest, and the spatial arrangements of the stands can influence the risk of a stand being wind-thrown. Thus, the selection of a forest management policy is an important economic matter. We propose to model the selection of a forest management policy as a GMDP. The graphical part of this example GMDP model is shown in Figure 1. The forest is divided into geographical contiguous parcels of land, called *stands*. Each stand is represented by a state variable (right part of the figure). Each stand variable represents the age of the stand, which can be used to describe both the timber value of the stand, and the protection from storms it provides to neighbour stands. Two actions can be applied to each stand node at each time step: "clear-cutting" and "do-nothing". An additional node (middle) represents the occurrence of a storm at a given time step. The dynamics of the process has a deterministic component linked to the natural age evolution, and to the clear-cutting actions (replacing all trees in a stand with new, "zero-aged", ones). It also has a stochastic component, linked to the probabilistic occurrence of a storm, which may damage stands and necessitate their (maybe unplanned) clear-cutting.

Now, the transition probabilities for a given stand depend on the action locally applied, and the occurrence of a storm. When a storm occurs, the probability of a stand being damaged is dependent on the state of the stand and the "shelter effect" provided by its neighbour stands. This since stands can block the wind and thereby decrease the risk of other stands being wind-thrown. The sheltering effect provided by a stand, depends on the age of the stand. It is assumed that the wind direction of the storms is always the same, in the considered area. Figure 1 shows the probabilistic dependencies between the state variables. Note that the neighbourhood relation is not symmetric, and that the size of a neighbourhood is linked to the number of parent nodes, not child nodes. Thus, even if the storm variable has many children (every stands), the size of its neighbourhood is zero⁴.

Finally, local rewards are attached to each stand and are obtained when "clear-cutting" is applied and increase with the age of the stand.

³ [2] claim that there always exist local policies which are optimal, however, the proof for this claim is wrong, and furthermore it is possible to find examples for which no local policy is globally optimal. Still, in practice, the best local policies often perform well.

⁴ We assume that storm probability is not influenced by the occurrence of a storm the previous year.

A fixed residual reward is obtained when "do-nothing" is applied. We consider than when a storm occurs, forced salvage harvest of damaged stands only benefits the compensate costs and does not change the residual reward.

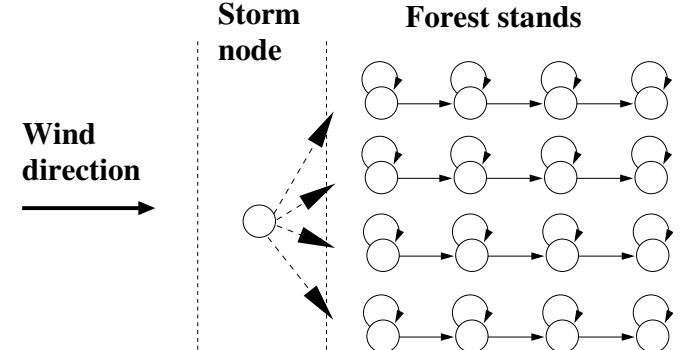


Figure 1. A GMDP representing the forest example, with one specific "storm event" node influencing all other stand-level state variable nodes.

4 APPROXIMATE LINEAR PROGRAMMING SOLUTION FOR GMDP

In this section we describe how the GMDP structure can be exploited for designing an efficient approximate linear programming algorithm. More precisely, we will show how the choice of specific basis functions can be used for finding an approximate solution of (4) in reasonable computation time. We will furthermore show how to compute a bound on the quality loss of the computed solution. We first describe a variation, providing a worst-case error bound, of the ALP approach that was presented in Section 2.

4.1 A generic ALP algorithm providing a bound on the measure of the worst-case error

A problem with the ALP algorithm described in Section 2 is that the quality of the approximate solution depends on the values chosen for the parameters $c(x)$. Furthermore, there are no guarantees that the approximate policy δ_{w^*} obtained from the solution w^* of the ALP, actually minimises the worst-case distance $\|v^* - v_{\delta_{w^*}}\|_\infty$.

With this problem in mind, [4] proposed an approximate *policy iteration* algorithm, in which they use an ALP method for computing an approximation of the value of a fixed policy. The algorithm explicitly minimises the worst-case error on the *measure* of the value of the policy. We suggest a variation of their algorithm which efficiently computes a set of parameter values w^* for which an *upper bound* on the worst-case distance to the optimal value function $\|v^* - v_{\delta_{w^*}}\|_\infty$ is minimised, instead of the worst-case distance itself.

Let us denote $\varepsilon_w = \|v^* - v_{\delta_w}\|_\infty = \max_{x \in \mathcal{X}} |v_{\delta_w}(x) - \mathcal{B}(v^*)(x)|$ the worst-case distance to the optimal value function associated to the parameters w of policy δ_w . Note that [8] showed that:

$$\|v^* - v\|_\infty \leq \frac{2\gamma}{1-\gamma} (\|v - \mathcal{B}(v)\|_\infty), \forall v : \mathcal{X} \rightarrow \mathbb{R}.$$

Using this result, we get the following bound on ε_w :

$$\varepsilon_w \leq \frac{2\gamma}{1-\gamma} (\|v_w - \mathcal{B}(v_w)\|_\infty). \quad (5)$$

Let $\phi_w = \|v_w - \mathcal{B}(v_w)\|_\infty$. Note that ϕ_w is called the *Bellman error* associated with v_w [8]. Furthermore, by finding the vector w which minimises ϕ_w we get a solution policy δ_{w^*} which approximates the optimal policy δ^* . Also, equation (5) provides us with an upper bound on ε_{w^*} , the worst-case distance from v_{w^*} to the optimal value function v^* .

The problem of minimising ϕ_w can be expressed by the following (non-linear) program:

$$\begin{aligned} \min \quad & \phi \\ \text{s.t.} \quad & \phi \geq v_w(x) - \mathcal{B}(v_w)(x), \forall x \in \mathcal{X} \\ & \phi \geq \mathcal{B}(v_w)(x) - v_w(x), \forall x \in \mathcal{X} \\ & w = (w_1, \dots, w_k) \text{ free}, \phi \text{ free}. \end{aligned} \quad (6)$$

4.2 Approximation in the GMDP case

Let $< \mathcal{X}, \mathcal{A}, p, r, G >$ be a GMDP, with $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$. In the following, the state space $\mathcal{X}_i \in \mathcal{X}$ of variable i is identified with the set $\{1, \dots, |\mathcal{X}_i|\}$ of integers.

Let us consider a parameterised approximation of the value function, similar to that of (3). Let us also define the basis functions in the GMDP framework as :

Definition 6 (GMDP basis functions) Let $< \mathcal{X}, \mathcal{A}, p, r, G >$ be a GMDP, with $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$. The basis functions: $h_{ij}, i \in 1, \dots, n, j \in 1, \dots, |\mathcal{X}_i|$ are defined as $h_{ij}(x_i) = 1$ if $x_i = j$ and 0 otherwise.

Notice that there are exactly $(\sum_{i=1}^n |\mathcal{X}_i|)$ basis functions. We can then write v_w as:

$$v_w(x) = \sum_{i=1}^n v_w^i(x_i) = \sum_{i=1}^n \sum_{j=1}^{|\mathcal{X}_i|} w_{ij} h_{ij}(x_i), \forall x \in \mathcal{X} \quad (7)$$

The Bellman transform $\mathcal{B}(v_w)$ can then be decomposed as:

Proposition 1 (GMDP Bellman operator)

$$\mathcal{B}(v_w)(x) = \sum_{i=1}^n \mathcal{B}_i(v_w^i)(x_{N(i)}), \forall x \in \mathcal{X}$$

$$\text{where: } \mathcal{B}_i(v_w^i)(x_{N(i)}) = \max_{a_i \in \mathcal{A}_i} \left\{ r_i(x_{N(i)}, a_i) + \gamma \sum_{x'_i \in \mathcal{X}_i} p_i(x'_i | x_{N(i)}, a_i) v_w^i(x'_i) \right\}, \forall x_{N(i)} \in \mathcal{X}_{N(i)}$$

Using the GMDP basis functions and the GMDP Bellman operator, (6) can be written as:

$$\begin{aligned} \min \quad & \phi \\ \text{s.t.} \quad & \phi \geq \sum_{i=1}^n (v_w^i(x_i) - \mathcal{B}_i(v_w^i)(x_{N(i)})), \forall x \in \mathcal{X} \\ & \phi \geq \sum_{i=1}^n (\mathcal{B}_i(v_w^i)(x_{N(i)}) - v_w^i(x_i)), \forall x \in \mathcal{X} \\ & w = (w_{11}, \dots, w_{n|\mathcal{X}_n|}) \text{ free}, \phi \text{ free} \end{aligned} \quad (8)$$

The NLP (8) is still intractable (it has $O(|\mathcal{X}| \times |\mathcal{A}|)$ constraints). We suggest to consider a first approximation of (8), in which the constraints that are of the form $\phi \geq |\sum_{i=1}^n \varphi_i(x_{N(i)})|$ are replaced by the following constraints :

Approximation 1:

$$\begin{aligned} \phi &= \sum_{i=1}^n \phi_i, \\ \phi_i &\geq |\varphi_i(x_{N(i)})|, \forall i, \forall x_{N(i)}. \end{aligned}$$

By making this approximation, the NLP (8) is replaced by a NLP that furthermore can be divided into n independent non-linear programs NLP_i , each of the following form:

$$\begin{aligned} \min \quad & \phi_i \\ \text{s.t.} \quad & \phi_i \geq v_w^i(x_i) - \mathcal{B}_i(v_w^i)(x_{N(i)}), \forall x_{N(i)} \in \mathcal{X}_{N(i)} \\ & \phi_i \geq \mathcal{B}_i(v_w^i)(x_{N(i)}) - v_w^i(x_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)} \\ & w_i = (w_{i1}, \dots, w_{i|\mathcal{X}_i|}) \text{ free}, \phi_i \text{ free} \end{aligned} \quad (9)$$

A tractable approximation of linear program (6) can then be found by solving the n independent problems (9). Each independent non-linear program NLP_i , only contains $|\mathcal{X}_i| + 1$ variables and $2 \times |\mathcal{X}_{N(i)}| \times |\mathcal{A}_i|$ constraints, in comparison with the original approximate NLP model (4), that with the proposed basis function contains $|\mathcal{X}| \times |\mathcal{A}|$ constraints, and $\sum_{i=1}^n |\mathcal{X}_i|$ variables.

The question that remains now is how are the n independent problems solved? Note that the second set of constraints of (9) can be expressed in the following form: $\phi_i \geq \max_{a_i \in \mathcal{A}_i} \psi(x_{N(i)}, a_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)}$. Which, equivalently can be expressed as: $\phi_i \geq \psi(x_{N(i)}, a_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i$. So, the second set of non-linear constraints in (9) can be replaced by an equivalent set of linear constraints.

The first set of constraints can be expressed as: $\phi_i \geq -\max_{a_i \in \mathcal{A}_i} \psi(x_{N(i)}, a_i)$, which is obviously not equivalent to: $\phi_i \geq -\psi(x_{N(i)}, a_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i$. Indeed, the second form is more restrictive than the first one, since it involves constraints which should be satisfied by every actions, while in the first form the constraints should be satisfied by at least one action. However, our second approximation consists in using this second form so as to recover a set of linear constraints.

Approximation 2:

By replacing the first set of constraints in the non-linear programs NLP_i (9) by the following more restrictive set of constraints: $\phi_i \geq -\psi(x_{N(i)}, a_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i$, we create new set of independent linear programs $LP_i, i \in 1, \dots, n$:

$$\begin{aligned} \min \quad & \phi_i \\ \text{s.t.} \quad & \phi_i \geq \sum_{j=1}^{|\mathcal{X}_i|} w_{ij} h_{ij}(x_i) - r_i(x_{N(i)}, a_i) \\ & -\gamma \sum_{x'_i \in \mathcal{X}_i} (p_i(x'_i | x_{N(i)}, a_i) \sum_{j=1}^{|\mathcal{X}'_i|} w_{ij} h_{ij}(x'_i)), \\ & \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i \\ & \phi_i \geq r_i(x_{N(i)}, a_i) - \sum_{j=1}^{|\mathcal{X}_i|} w_{ij} h_{ij}(x_i) \\ & + \gamma \sum_{x'_i \in \mathcal{X}_i} (p_i(x'_i | x_{N(i)}, a_i) \sum_{j=1}^{|\mathcal{X}'_i|} w_{ij} h_{ij}(x'_i)), \\ & \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i \\ & w_{ij} \text{ free, for } j = 1, \dots, |\mathcal{X}_i|, \phi_i \text{ free} \end{aligned}$$

Which can be written as:

$$\begin{aligned} \min \quad & \phi_i \\ \text{s.t.} \quad & \phi_i \geq \sum_{j=1}^{|\mathcal{X}_i|} w_{ij} (h_{ij}(x_i) - \gamma p_i(x'_i = j | x_{N(i)}, a_i)) \\ & - r_i(x_{N(i)}, a_i), \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i \\ & \phi_i \geq r_i(x_{N(i)}, a_i) - \sum_{j=1}^{|\mathcal{X}_i|} w_{ij} (h_{ij}(x_i) \\ & - \gamma p_i(x'_i = j | x_{N(i)}, a_i)), \forall x_{N(i)} \in \mathcal{X}_{N(i)}, \forall a_i \in \mathcal{A}_i \\ & w_{ij} \text{ free, for } j = 1, \dots, |\mathcal{X}_i|, \phi_i \text{ free} \end{aligned}$$

Note that in the expression of the LP_i , the parameters w_{ij} now appear explicitly. The advantage of this approach is that the solution $\tilde{w}_i = (\tilde{w}_{i1}, \dots, \tilde{w}_{i|\mathcal{X}_i|})$ of each subsystem LP_i , can be used to determine the parameter values of $\tilde{w} = (\tilde{w}_1, \dots, \tilde{w}_n)$. This in turn, defines a parameterised approximation $v_{\tilde{w}}$ of the optimal value function v^* . Another advantage of this approximation is that the greedy policy $\tilde{\delta}$ associated with $v_{\tilde{w}}$ is local, since the following holds for all

$x \in \mathcal{X}$:

$$\tilde{\delta}(x) = \arg \max_{a \in \mathcal{A}} \left\{ \sum_{i=1}^n r_i(x_{N(i)}, a_i) + \gamma \sum_{x'_i \in \mathcal{X}_i} p_i(x'_i | x_{N(i)}, a_i) v_{\tilde{w}}^i(x'_i) \right\}$$

where $v_{\tilde{w}}^i$ are defined as in (7). So, the local components $(\tilde{\delta}_1(x_{N(1)}), \dots, \tilde{\delta}_n(x_{N(n)}))$ of $\tilde{\delta}$ can be computed as:

$$\forall i \in 1, \dots, n, \forall x_{N(i)} \in \mathcal{X}_{N(i)}$$

$$\begin{aligned} \tilde{\delta}_i(x_{N(i)}) &= \arg \max_{a_i \in \mathcal{A}_i} \left\{ r_i(x_{N(i)}, a_i) + \right. \\ &\quad \left. \gamma \sum_{x'_i \in \mathcal{X}_i} p_i(x'_i | x_{N(i)}, a_i) v_{\tilde{w}}^i(x'_i) \right\}. \end{aligned}$$

Thus, solving the n linear programs allows a computation of the approximate local policy $\tilde{\delta}$ which is an approximation of δ^* . A bound on the worst case error $\|v^* - v_{\tilde{\delta}}\|_\infty$ can of course easily be computed by applying (5).

5 EXPERIMENTAL RESULTS

In order to assess the new Approximate Linear Programming algorithm, it was tested on the wind-throw problem described in section 3.2. The tests were performed with increasing number n of stands in the forest. The value of $|\mathcal{X}_i|$ and $|\mathcal{A}_i|$ were fixed to $|\mathcal{X}_i| = 4$ and $|\mathcal{A}_i| = 2, \forall i = 1, \dots, n, \forall i = 1, \dots, n$. The cpu run times of the ALP algorithm and an exact Policy Iteration (PI) algorithm (when available), for increasing values of n can be seen in table 1:

Table 1

n	ALP	PI
4	0.78	2151.23
9	2.09	
16	4.03	
25	6.54	
36	9.69	

In order to asses the policy obtained by the ALP algorithm, it was compared to the exact PI algorithm, a heuristic *greedy* policy⁵, and a heuristic *random* policy⁶. As for large problems, the computation of the value function of a specific policy is very time consuming. The value functions of the policies were evaluated using Monte Carlo simulations. The Monte Carlo simulations were used to compute the average value of a policy, for a number of randomly chosen starting points. The average value of the different policies for increasing values of n can be seen in Table 2:

Table 2

n	ALP	PI	Greedy Policy	Random Policy
4	45.23	45.37	32.20	17.79
9	109.12		70.14	44.08
16	192.47		120.81	81.06
25	295.68		186.06	127.95
36	418.58		266.12	182.82

These tests show the advantage of our new ALP algorithm as for fixed sizes of the neighbourhood function, the run times only increase linearly in the number of stands, n . This while generating a better approximation of the optimal policy than both heuristic greedy policy and heuristic random policy and very close to the exact PI results, when available.

⁵ The greedy policy is local and simply consists in taking in each node the action $\delta_i(x_{N(i)}) = a_i^*$ maximising $r_i(x_{N(i)}, a_i)$.

⁶ The random policy is local and simply consists in taking in each node a random action.

6 CONCLUDING REMARKS

In this paper, we have described a specific form of factored MDP (FMDP) which uses a graph-based representation of local dependencies: the GMDP framework. An important feature of GMDP is that they admit problem and approximately optimal solution representations in space linear in the number n of nodes of the underlying graph. We have proposed an *approximate linear programming* (ALP) algorithm which, by a simplification of the spatio-temporal dependence structure of the model, requires only linear space and time, provided that the maximum number of neighbours remains small. Experiments on toy models inspired by real life decision problems showed that the loss in quality of the policy returned is limited. In particular, the approximate algorithm greatly improves the greedy short-term solution which is usually the only one available in problems of high dimension.

Our approach should be contrasted to that of [1, 4] which can only handle problems with a small number of decision variables and are thus inadequate to solve GMDP. One exception is [5], in which *collaborative multi-agent factored MDP* are studied (chap. 8 and 9) and a dedicated approximation algorithm is proposed. The empirical comparison between this algorithm and ours has not been performed yet, but the complexity of this algorithm is quadratic in the number of decision variables while ours is linear. Note still that *collaborative multi-agent factored MDP* are more general than GMDP. [3] also propose a method to tackle FMDP with multiple decision variables based on the idea of sampling the (exponential size) set of constraints generated by a linear programming (LP) modelling of a FMDP. However, to keep acceptable probabilistic bounds on the quality of the returned policy, the sample size should remain proportional to the number of possible actions, i.e. exponential in the number of decision variables. [5] points out experiments in which his method outperforms that of [3] when the same computation time is allowed to both methods. Finally, [6] propose an alternative *Approximate Policy Iteration* method for solving GMDP, based on the *mean-field approximation* of a stochastic process *occupation measure*. Their method is also of complexity linear in the number of variables. The experimental studies led so far show that even if the two approaches are different, the returned policies have similar values, and the CPU times are linearly related.

REFERENCES

- [1] C. Boutilier, R. Dearden, and M. Goldszmidt, ‘Stochastic dynamic programming with factored representations’, *Artificial Intelligence*, **121**(1), 49–107, (2000).
- [2] R. K. Chornei, H. Daduna, and P. S. Knopov, ‘Controlled markov fields with finite state space on graphs’, *Stochastic Models*, **21**, 847–874, (2005).
- [3] D. P. de Farias and B. Van Roy, ‘On constraint sampling in the linear programming approach to approximate dynamic programming’, *Math. of Op. Research*, **29**(3), 462–478, (2004).
- [4] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, ‘Efficient solution algorithms for factored MDPs’, *Journal of Artificial Intelligence Research*, **19**, 399–468, (2003).
- [5] C. E. Guestrin, *Planning under uncertainty in complex structured environments*, Ph.D. dissertation, Stanford University, 2003.
- [6] N. Peyrard and R. Sabbadin, ‘Mean field approximation of the policy iteration algorithm for graph-based mdp’, in *Proceedings ECAI’06*, (2006).
- [7] M. L. Puterman, *Markov Decision Processes*, John Wiley and Sons, New York, 1994.
- [8] R. J. Williams and L.C.I. Baird, ‘Tight performance bounds on greedy policies based on imperfect value functions’, Technical report, College of Computer Science, Northeastern University, Boston, MA, (1993).

Mean Field Approximation of the Policy Iteration Algorithm for Graph-based Markov Decision Processes

Nathalie Peyrard¹ and Régis Sabbadin²

Abstract. In this article, we consider a compact representation of multidimensional Markov Decision Processes based on Graphs (GMDP). The states and actions of a GMDP are multidimensional and attached to the vertices of a graph allowing the representation of local dynamics and rewards. This approach is in the line of approaches based on *Dynamic Bayesian Networks*. For policy optimisation, a direct application of the *Policy Iteration* algorithm, of exponential complexity in the number of nodes of the graph, is not possible for such high dimensional problems and we propose an approximate version of this algorithm derived from the GMDP representation. We do not try to approximate directly the value function, as usually done, but we rather propose an approximation of the *occupation measure* of the model, based on the *mean field* principle. Then, we use it to compute the value function and derive approximate policy evaluation and policy improvement methods. Their combination yields an approximate *Policy Iteration* algorithm of linear complexity in terms of the number of nodes of the graph. Comparisons with the optimal solution, when available, and with a naive short-term policy demonstrate the quality of the proposed procedure.

1 Introduction

Markov Decision Processes (MDP, [9]) have become a standard model for decision-theoretic planning. However, they reach their limits when dealing with spatial applications (e.g. control in ecology, epidemiology, computer networks...). In such contexts, the size of the problem induces two difficulties: the representation and the time complexity for the MDP resolution. To circumvent the first one, factored representations have been developed. Several methods of aggregation/decomposition for large problems have been proposed in the AI community. These methods ([2, 7]) use a compact representation of probability and reward functions involved in the MDP description and propose algorithms adapted for these factored descriptions. However, these representations and algorithms usually only consider the structure of the states space, not of the actions space. In this article, we exploit a particular form of compact representation of MDP, based on graphs modelling both states and actions spaces in a common structure, that we will refer to as Graph MDP (GMDP). In this framework, several states and actions variables are attached to the vertices of a graph allowing the representation of local dependencies. The dynamics of each state variable only depends on the action applied locally and on the current values of the neighbouring state variables. The classical *Policy Iteration* algorithm for policy optimisation does not scale to the size of factored MDP or GMDP: the time complexity is exponential in the number of nodes in the graph.

¹ INRA, Avignon - France, email: peyrard@avignon.inra.fr

² INRA, Toulouse - France, email: sabbadin@toulouse.inra.fr

We propose an approximate version of the *Policy Iteration* algorithm derived from the GMDP representation and from a *mean field* approximation of the stochastic model, leading to an approximation of the *occupation measure* of the model. Then, we propose approximate policy evaluation and policy improvement steps, based on the occupation measure. Their combination yields an approximate *Policy Iteration* algorithm and we discuss the computational complexity of the proposed method. Then we assess its performance in terms of precision by comparing experimentally the method to the exact *Policy Iteration* algorithm when possible, and to a greedy policy when not, on toy examples.

2 Graph-Based Markov Decision Processes

2.1 Definitions

In its classical formulation [9], a finite stationary MDP is described by a four-tuple $\langle \mathcal{X}, \mathcal{A}, p, r \rangle$, where \mathcal{X} represents the finite set of admissible states of the system, \mathcal{A} the finite set of applicable actions, $p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$ the transition probabilities between states and $r : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ an “immediate” reward function. In a MDP, the state $x' \in \mathcal{X}$ at step $t + 1$ depends only on the state $x \in \mathcal{X}$ at time t and on the action $a \in \mathcal{A}$ applied at this instant (Markov property). The transition probability from x to x' given a is $p(x'|x, a)$. In addition, a *reward* function r is defined as the reward obtained when action a is applied on the system in the current state. When the process is stationary, both $p(x'|x, a)$ and $r(x, a)$ are independent of t . In this article, we consider the situation where the state $x \in \mathcal{X}$ is multidimensional, and the coordinates are not independent. They are locally interacting and the interaction network can be represented by a graph. The transition probabilities and the rewards are local according to the graph structure. This representation has already been exploited in [4]: a GMDP is defined by a 5-tuple $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$, the state space is a Cartesian product $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and the action space is a Cartesian product $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$. $G = (V, E)$ is an oriented graph, defined by a set of vertices $V = \{1, \dots, n\}$ and a set of (oriented) edges $E \subseteq V^2$. An edge (i, j) means that node i influences node j (i is a *parent* of j). A neighbourhood function N is defined over V by the set of parents of a given node :

Definition 1 (Neighbourhood function)

$N : V \rightarrow 2^V$ is defined by : $\forall i \in V, N(i) = \{j \in V, (j, i) \in E\}$.

We also introduce the following characteristics of the GMDP: $\sigma = \max_i |\mathcal{X}_i|$, $\alpha = \max_i |\mathcal{A}_i|$ and $\nu = \max_i |N(i)|$ (ν is the maximum degree of a node in the graph).

In a GMDP, transition probabilities and rewards are local according to G :

Definition 2 (Local transitions) Let $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$ be a GMDP. Transitions are said to be local iff for all $x = (x_1 \dots x_n), x' = (x'_1 \dots x'_n) \in \mathcal{X}, a = (a_1 \dots a_n) \in \mathcal{A}$,

$$p(x'|x, a) = \prod_{i=1}^n p_i(x'_i|x_{N(i)}, a_i),$$

where $\forall I \subseteq \{1, \dots, n\}$, $x_I = \{x_i\}_{i \in I}$. With this factored representation, the space complexity of the representation of p is now $O(n \cdot \sigma^{\nu+1} \cdot \alpha)$, instead of $O((\sigma^2 \cdot \alpha)^n)$ for a classical MDP.

Definition 3 (Local rewards) Let $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$ be a GMDP. Rewards are said to be local when $\forall x = (x_1 \dots x_n) \in \mathcal{X}, \forall a = (a_1 \dots a_n) \in \mathcal{A}$,

$$r(x, a) = \sum_{i=1}^n r_i(x_{N(i)}, a_i).$$

A function $\delta : \mathcal{X} \rightarrow \mathcal{A}$ assigning an action to every state is called a policy. In the general case, policies for a GMDP take the form $\delta = (\delta_1, \dots, \delta_n)$, where $\delta_i : \mathcal{X} \rightarrow \mathcal{A}_i$. Nevertheless, global policies can take space in $O(n \cdot \sigma^n)$ at most to be expressed, which can be prohibitive, except for very low dimensionality problems. Some special policies, called *local policies* are of special interest since they take space in $O(n \cdot \sigma^\nu)$.

Definition 4 (Local policy) In a GMDP $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$, a policy $\delta : \mathcal{X} \rightarrow \mathcal{A}$ is said to be local iff $\delta = (\delta_1, \dots, \delta_n)$ where $\delta_i : \mathcal{X}_{N(i)} \rightarrow \mathcal{A}_i$.

2.2 GMDP example

Let us consider the graph in Figure 1, representing crop fields infected or not by a disease. Each node i , representing a field, has two possible states : $x_i = 1$ if uninfected and $x_i = 2$ if infected ($\mathcal{X}_i = \{1, 2\}$). Edges represent possible paths for contamination from neighbour locations. The disease passes through an edge with fixed probability p (in this case the graph is non-oriented). In addition, every location has probability ε of being infected through long range contamination (from another region). Decisions are taken yearly and two actions are possible for each node : $a_i = 1$ if a normal culture mode is used, and $a_i = 2$ if the field is left fallow and treated ($\mathcal{A}_i = \{1, 2\}$). Yields are affected by the state of the field and the applied action: the disease halves the yearly yield, while leaving the crop fallow generates no yield. Transition probabilities and rewards are summarised in Table 1. The objective in such context is to optimise the choice of a long-term policy in terms of expected yield.

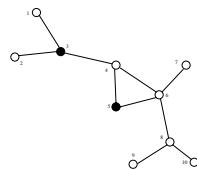


Figure 1. An epidemiological problem state: white (resp. black) nodes represent uninfected (resp. infected) fields.

2.3 Search for an optimal policy

Let $\langle \mathcal{X}, \mathcal{A}, p, r, G \rangle$ be a GMDP, discounted with discount factor γ . In the stationary case, the problem of searching an optimal policy

		$p(x'_i x_{N(i)}, a_i)$			
		$a_i = 1$		$a_i = 2$	
		$x_i = 1$	$x_i = 2$	$x_i = 1$	$x_i = 2$
$x'_i = 1$		$1 - P(\varepsilon, p)$	0	1	q
$x'_i = 2$		$P(\varepsilon, p)$	1	0	$1 - q$

		$r(x_i, a_i)$	
		$a_i = 1$	$a_i = 2$
		$x_i = 1$	$x_i = 2$
$x_i = 1$		r	0
$x_i = 2$		$r/2$	0

Table 1. Epidemiological problem: transition probabilities and rewards, $P(\varepsilon, p) = \varepsilon + (1 - \varepsilon)(1 - (1 - p)^{n_i})$, with n_i the number of parents of i which are infected.

with respect to the infinite horizon-discounted value of a policy δ , $V_\delta(x) = E \left[\sum_{t=0}^{+\infty} \gamma^t r(X^t, \delta(X^t)) \mid X^0 = x \right]$, can be written as:

$$\text{Find } \delta^*, \mathcal{X} \rightarrow \mathcal{A}, V_{\delta^*}(x) \geq V_\delta(x), \forall x \in \mathcal{X}, \forall \delta \in \mathcal{A}^\mathcal{X}$$

(NB: In this section and in the following, we will use upper case letters to represent random variables, and lower case ones for realisations of these random variables.)

This problem is classically solved by Stochastic Dynamic Programming methods [9] such as the *Backwards Induction*, *Policy Iteration* and *Value Iteration* algorithms.

The *Policy Iteration* (PI) algorithm consists in the alternation of an *evaluation* phase of the current policy and an *improvement* phase of this policy.

Policy Evaluation step: Solve

$$V_\delta(x) = r(x, \delta(x)) + \gamma \cdot \sum_{x' \in \mathcal{X}} p(x'|x, \delta(x)) \cdot V_\delta(x'), \forall x.$$

Policy improvement step: Update the current policy δ into a policy δ' by applying for all x :

$$\delta'(x) = \arg \max_{a \in \mathcal{A}} \left(r(x, a) + \gamma \cdot \sum_{x' \in \mathcal{X}} p(x'|x, a) \cdot V_\delta(x') \right).$$

It is guaranteed that $V_{\delta'}(x) \geq V_\delta(x), \forall x$, after the improvement phase. Furthermore when $V_{\delta'}(x) = V_\delta(x), \forall x$ we have the guarantee that δ is an optimal policy [9].

Generally, the PI algorithm converges within a very small number of iterations, but each iteration is costly since policy evaluation needs solving a system of equations (complexity in $O(\sigma^{3n})$) and policy improvement has complexity in $O((\sigma^2 \cdot \alpha)^n)$. This exponential complexity limits severely the size of GMDP that can be efficiently solved. In this paper we propose an approximation of the two steps of the PI algorithm for a GMDP, based on the search of an (a priori sub-optimal) policy among the local ones, which leads to a time complexity linear in n (but exponential in ν).

3 Approximate PI for GMDP

3.1 Policy Value and Occupation Measures

There exists an alternative way of expressing a policy value, based on the occupation measure, which has been used for instance in [1]. The occupation measure is related to the time a stochastic process spends in a given state, knowing the initial state. By extension, occupation measures have also been used in the framework of discounted MDP.

Definition 5 (Occupation measure) Let $\langle \mathcal{X}, \mathcal{A}, p, r \rangle$ be a stationary MDP, γ a discount factor. Let $\delta : \mathcal{X} \rightarrow \mathcal{A}$ be a policy and

$x^0 \in \mathcal{X}$ an initial state. The occupation measure $P_{x^0, \delta, \gamma} : \mathcal{X} \rightarrow [0, 1]$ is defined as:

$$\forall y \in \mathcal{X}, P_{x^0, \delta, \gamma}(y) = (1 - \gamma) \sum_{t=0}^{+\infty} \gamma^t \cdot P_\delta(X^t = y | X^0 = x^0).$$

The value function V_δ of any policy can be computed from the probability distribution $P_{x^0, \delta, \gamma}$ as

$$\forall x^0 \in \mathcal{X}, V_\delta(x^0) = \frac{1}{1 - \gamma} \cdot \sum_{y \in \mathcal{X}} P_{x^0, \delta, \gamma}(y) \cdot r(y, \delta(y)).$$

In particular, $\forall \delta$ a local policy, the value function can be decomposed as $V_\delta(x^0) = \sum_{i=1}^n V_\delta^i(x^0)$, where

$$V_\delta^i(x^0) = \sum_{t=0}^{+\infty} \gamma^t \cdot \sum_{y \in \mathcal{X}} P_\delta(X^t = y | X^0 = x^0) \cdot r_i(y_{N(i)}, \delta_i(y_{N(i)})). \quad (1)$$

As such, this expression does not help to compute efficiently optimal or near optimal policies. However, we propose to look for an approximation \hat{C} of the conditional probability $P_\delta(X^t = y | X^0 = x^0)$ with simpler structure, which will result in an approximation of $V_\delta^i(x^0)$ by a function depending only on $x_{N(i)}^0$. Conditional probabilities (from time 0 to time t) can be derived from the transition probabilities $p_i(x_i^t | x_{N(i)}^{t-1}, \delta(x_{N(i)}^{t-1}))$. If we are able to find approximate transition probabilities of the form $\hat{Q}_\delta^{t,i}(x_i^t | x_i^{t-1})$ then the corresponding approximate conditional probability will be, $\forall t \geq 2$,

$$\begin{aligned} \hat{C}_\delta^t(y|x^0) &= \sum_{x^{t-1}} \hat{Q}_\delta^t(y|x^{t-1}) \cdot \hat{C}_\delta^{t-1}(x^{t-1}|x^0) \\ &= \prod_{i=1}^n \sum_{x_i^{t-1}} \hat{Q}_\delta^{t,i}(y_i|x_i^{t-1}) \cdot \hat{C}_\delta^{t-1,i}(x_i^{t-1}|x_i^0). \end{aligned} \quad (2)$$

the recursive definition being initialised by $\hat{C}_\delta^1(y|x^0) = \prod_i \hat{Q}_\delta^{1,i}(y_i|x_i^0)$. And by induction, $\forall t, \hat{C}_\delta^t(y|x^0) = \prod_{i=1}^n \hat{C}_\delta^{t,i}(y_i|x_i^0)$, so that we get $V_\delta^i(x^0) \approx \hat{V}_\delta^i(x_{N(i)}^0)$ where

$$\hat{V}_\delta^i(x_{N(i)}^0) = \sum_{t=0}^{+\infty} \gamma^t \cdot \sum_{y_{N(i)}} \left(\prod_{j \in N(i)} \hat{C}_\delta^{t,j}(y_j|x_j^0) \cdot r_i(y_{N(i)}, \delta_i(y_{N(i)})) \right).$$

The main contribution of this work is to propose a solution for the choice of $\hat{Q}_\delta^{t,i}(x_i^t | x_i^0)$, derived from the mean field principle, as detailed in the next section.

3.2 Mean field approximation of the transition probabilities

The mean field principle arises from statistical mechanics [3] and the study of systems of particles in interaction and Gibbs distributions. It has since been successfully used in other domains such as image analysis, graphical models, epidemiology, ecology. The mean field approximation of a multidimensional distribution $P(u_1, \dots, u_m)$ is defined as the best approximation by a system of independent variables in the sense of the Kullback-Leibler divergence, $KL(Q|P) = E_Q[\log(Q/P)]$. The minimum of $KL(Q|P)$ is thus looked for among the distributions Q with factorisation property: $Q(u_1, \dots, u_m) = \prod_{i=1}^m Q_i(u_i)$.

In the GMDP framework, the model complexity lies in the spatio-temporal dependence between sites' states. More specifically, if

$X^t = \{X_1^t, \dots, X_n^t\}$ is the random variable corresponding to the system's state at time t then, given a policy δ , the variable X_i^t depends on the variables in its neighborhood at the previous time step, $X_{N(i)}^{t-1}$ through the transition probabilities. So, we propose to apply the mean field principle to the joint distribution of X^{t-1} and X^t , which is fully defined by the distribution of X^{t-1} , and the transition probability of X^t knowing X^{t-1} .

Let us start the iterative approximation procedure at time $t = 1$. We will use the notation P for the exact distributions and the notation Q for the approximate ones. Assuming an initial, factored, distribution P^0 on \mathcal{X}^0 (for sake of simplicity and also to ensure the repeatability of the approximation method from one time step to the next one) we have the following forms for the exact and the approximate distributions:

$$\begin{aligned} P^0(x^0) &= \prod_{i=1}^n P^{0,i}(x_i^0), & P_\delta(x^1|x^0) &= \prod_{i=1}^n p_i(x_i^1|x_{N(i)}^0, \delta_i(x_{N(i)}^0)) \\ Q^0(x^0) &= P^0(x^0), & Q_\delta^1(x^1|x^0) &= \prod_{i=1}^n Q_\delta^{1,i}(x_i^1|x_i^0) \end{aligned} \quad (3)$$

With the approximate model, given δ , $X^{1,i}$ depends only on $X^{0,i}$. Let us denote \mathcal{Q} the set of all joint distributions of X^{t-1} and X^t which can be decomposed as (3). Among all the elements of \mathcal{Q} , we are looking for the best approximation of the exact joint distribution, according to the Kullback-Leibler divergence. Since $Q^0 = P^0$, the approximation is only on Q_δ^1 .

Proposition 1 (Mean field approximation of the transitions) The mean field approximation \hat{Q}_δ^1 , defined as the solution of

$$\hat{Q}_\delta^1 = \arg \min_{Q_\delta^1 \in \mathcal{Q}} KL(Q_\delta^1(X^1|X^0).P^0(X^0)|P_\delta(X^1|X^0).P^0(X^0))$$

is

$$\hat{Q}_\delta^{1,i}(x_i^1|x_i^0) \propto \exp E_{P^0} [\log p_i(x_i^1|x_i^0, X_{N(i)\setminus\{i\}}^0, \delta_i(x_i^0, X_{N(i)\setminus\{i\}}^0))] \quad (4)$$

Proof: Indeed, we have

$$\begin{aligned} &KL(Q_\delta^1(X^1|X^0).P^0(X^0)|P_\delta(X^1|X^0).P^0(X^0)) \\ &= \sum_{x^0, x^1 \in \mathcal{X}^2} Q_\delta^1(x^1|x^0).P^0(x^0). \log \frac{Q_\delta^1(x^1|x^0)}{P_\delta(x^1|x^0)} \end{aligned}$$

Setting the derivates according to the $Q_\delta^{1,i}(x_i^1|x_i^0)$ to zero, we obtain the mean field solution. \square

By switching the expectation and the logarithm, we obtain the approximation

$$\hat{Q}_\delta^{1,i}(x_i^1|x_i^0) = E_{P^0} [p_i(x_i^1|x_i^0, X_{N(i)\setminus\{i\}}^0, \delta_i(x_i^0, X_{N(i)\setminus\{i\}}^0))]$$

It is easy to check that this quantity is normalized. It is easier to compute and more intuitive than expression (4). We will use this approximation in the following as the mean field approximation of the transition probabilities. Note that this factored expression relies on an initial factored a priori distribution P^0 . This initial distribution is necessary for technical reasons, but can also help representing a priori knowledge about the initial distribution of the process. If no a priori knowledge is available, we simply chose P^0 as a product of independent uniform laws on the \mathcal{X}_i 's.

Given the approximate transition probabilities between time 0 and

time 1, and given P^0 , we can derive the approximate probability distribution of X^1 for a given policy δ :

$$\hat{Q}_\delta^1(x^1) = \sum_{x^0} \hat{Q}_\delta^1(x^1|x^0).P^0(x^0) = \prod_{i=1}^n \sum_{x_i^0} \hat{Q}_\delta^{1,i}(x_i^1|x_i^0).P^{0,i}(x_i^0)$$

This approximate distribution at time 1 has also the factorisation property. Thus we can iterate the approximation method (minimisation of the Kullback-Leibler divergence) and obtain for each time step $t \geq 2$ an approximation of the transition probabilities, $\hat{Q}_\delta^{t+1}(x^{t+1}|x^t)$. Note that they depend on t (and on P^0) while this is not the case in the initial model.

3.3 Approximate Policy Evaluation

Using the mean field approximation of the occupation measure to perform the evaluation step, we recall that we obtain an approximation of each component of the policy value of the following form:

$$\begin{aligned} \hat{V}_\delta^i(x) &\approx \hat{V}_\delta^i(x_{N(i)}) \\ &\approx \sum_{t=0}^{+\infty} \gamma^t \cdot \sum_{y_{N(i)}} \left(\prod_{j \in N(i)} \hat{C}_\delta^{t,j}(y_j|x_j) \right) \cdot r_i(y_{N(i)}, \delta_i(y_{N(i)})) \end{aligned}$$

In this approximation, each component \hat{V}_δ^i depends on $x_{N(i)}$ only. This results in a reduced time complexity, exponential in the maximum number of neighbours of a node, as opposed to exponential in the number of nodes for the exact PI.

Proposition 2 (Complexity of the Approximate Policy Evaluation) *Let ϵ be the stopping threshold in the computation of the infinite sum over time. Then the complexity of the Approximate Policy Evaluation step is in $O(\ln(1/\epsilon) \cdot n \cdot \sigma^{3\nu} \cdot \nu^2)$.*

Proof: There are $n \cdot \sigma^\nu$ quantities $\hat{V}_\delta^i(x_{N(i)})$ to compute. Then, for a given site i and a given configuration $x_{N(i)}$, the computation of $\gamma^t \sum_{y_{N(i)}} \left(\prod_{j \in N(i)} \hat{C}_\delta^{t,j}(y_j|x_j) \right) \cdot r_i(y_{N(i)}, \delta_i(y_{N(i)}))$ requires at most σ^ν summations, ν products, and the evaluation of $\hat{C}_\delta^{t,j}(y_j^t|x_j)$. This last evaluation is of complexity $\nu \cdot \sigma \cdot \sigma^{\nu-1}$: sum over each possible state of site j at the previous time step and computation of the approximated transition probabilities as expectations of products of $\nu+1$ terms. The discount factor γ^t ensures that the precision required is reached for a time $t > K \frac{\ln(\epsilon)}{\ln(\gamma)}$. Finally, the complexity of the Approximate Policy Evaluation is $O(\ln(1/\epsilon) \cdot n \cdot \sigma^{3\nu} \cdot \nu^2)$. \square

3.4 Approximate Policy Improvement

In order to reduce the search space, we limit the search of approximately optimal policies among local ones (even if in general there is no guarantee that there exist a local policy which is optimal). This allows a reduction in representation space and in time complexity. Using the mean field approximation of V_δ and the local properties of rewards and transition probabilities, the exact maximisation

$$\delta'(x) = \arg \max_{a \in \mathcal{A}} \left(r(x, a) + \gamma \cdot \sum_{y \in \mathcal{X}} p(y|x, a) \cdot V_\delta(y) \right), \forall x.$$

is replaced by the search of the arguments of the maximum of the expressions :

$$\sum_i r_i(x_{N(i)}, a_i) + \gamma \sum_{y_{N(i)}} \left(\prod_{j \in N(i)} p_j(y_j|x_{N(j)}, a_j) \hat{V}_\delta^i(y_{N(i)}) \right).$$

Let us consider a particular site i . The terms which depend on a_i are

$$r_i(x_{N(i)}, a_i) + \gamma \sum_{k \in N(i)} \sum_{y_{N(k)}} \left(\prod_{j \in N(k)} p_j(y_j|x_{N(j)}, a_j) \cdot \hat{V}_\delta^i(y_{N(k)}) \right).$$

In order to ensure that the argument of the maximum is local, first we adopt a parallel scheme, initialised by δ , to compute each component of the improved policy:

$$\begin{aligned} \forall i, \tilde{\delta}_i^{n+1}(x) &= \arg \max_{a_i \in \mathcal{A}_i} r_i(x_{N(i)}, a_i) + \gamma \sum_{k \text{ s.t. } i \in N(k)} \sum_{y_{N(k)}} \left[\right. \\ &\quad \left. \left(\prod_{j \in N(k), j \neq i} p_j(y_j|x_{N(j)}, \tilde{\delta}_j^n(y_{N(j)})) \right) \cdot p_i(y_i|x_{N(i)}, a_i) \cdot \hat{V}_\delta^i(y_{N(k)}) \right]. \end{aligned}$$

This is not enough since the i th component of the solution still depend on $x_{N(N(i))}$. So, we replace the transition probabilities p_j by appropriate mean values, in order to restrict dependencies, as follows

$$\begin{aligned} p_i(y_i|x_{N(i)}, a_i) &\approx \frac{\sum_{x_{N(i)} \setminus \{i\}} p_i(y_i|x_{N(i)}, a_i)}{\text{card}(\mathcal{X}_{N(i)} \setminus \{i\})} \\ \text{if } j \in N(i): \\ p_j(y_j|x_{N(j)}, \delta(x_{N(j)})) &\approx \frac{\sum_{x_{N(j)} \setminus \{j\}} p_j(y_j|x_{N(j)}, \delta_j(x_{N(j)}))}{\text{card}(\mathcal{X}_{N(j)} \setminus \{j\})} \\ \text{if } j \notin N(i): \\ p_j(y_j|x_{N(j)}, \delta(x_{N(j)})) &\approx \frac{\sum_{x_j} p_j(y_j|x_{N(j)}, \delta_j(x_{N(j)}))}{\text{card}(\mathcal{X}_j)} \end{aligned}$$

Proposition 3 (Complexity of Approximate Policy Update) *The Approximate Policy Update step is of complexity $O(n \cdot \alpha \cdot \nu^2 \cdot \sigma^{2\nu})$.*

Proof: The maximisation is performed for all sites, and is over α terms at most. The computation of the approximate version of expression (5) requires at most $\nu \cdot \sigma^\nu$ summations and ν products. One element of the product is computed either in σ^ν or in σ operations. This leads to a complexity of at most $O(n \cdot \alpha \cdot \nu^2 \cdot \sigma^{2\nu})$. \square

3.5 Experimental evaluation of API

We first tested the mean field approximation of the Policy Iteration (MF-PI) algorithm on problems of small size, where the exact solution of the PI algorithm for standard MDP is available, in order to assess the quality of the approximate policies returned by MF-PI. The first toy example used is the one presented in Section 2.2. The parameters were set to $p = 0.2, \epsilon = 0.01, q = 0.9$. The second example is an extension to four states per node: the state 'healthy' and three states of increasing severity in the case of an infected field.

Example 1				Example 2			
n	error	CPU MF	CPU PI	n	error	CPU MF	CPU PI
3	0%	1.44	10.11	3	0%	21.4	646
4	6.5%	5.75	65.52	4	3.6%	574	21200
5	3.3%	6.78	454.54	40	4380		
6	10.1%	9.07	3485.6				

Table 2. Mean relative error (in percentage) between the optimal policy and the estimated one, and computation time.

In both cases, the graphs were generated randomly, with a maximum size of $N(i)$ of 3. For varying values of n , we observed a limited loss in the quality of the optimal policy (with a mean relative error no more than 10 %, see Table 2) and a significant gain in time complexity (see Figure 2). We observe a linear progression of the CPU time required by MF-PI with respect to the number of nodes (see Figure 2). Implementation was done in Scilab, on a quadriprocessor PC with four Intel 2.40 GHz processors with 512 Ko of cache.

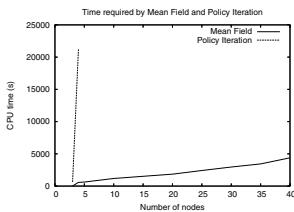


Figure 2. CPU times required by MF-PI and by exact PI

In a second set of experiments, we compared the relative performance of the MF-PI and the greedy (maximising the local immediate rewards) policies for graphs of larger size, on the problem of epidemiology control (version with three states of disease severity). Graph of dependencies were generated randomly and the MF-PI and the greedy policies were computed and their values were estimated using Monte Carlo simulations. Figure 3 illustrates the gain in the empirical value using MF-PI instead of the greedy policy, as a average over 5 problems, 10 initial states per problem, and simulation of 100 trajectories of length 20.

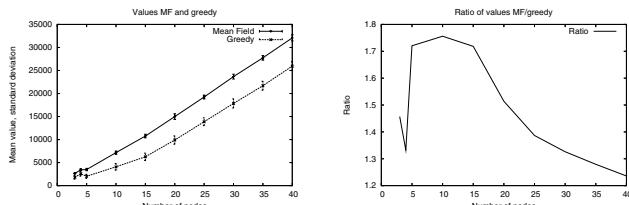


Figure 3. Left: estimated mean values of the MF-PI and the greedy policies, Right: estimated mean value of the ratio of the MF-PI policy value over the greedy policy value.

4 Concluding remarks

In this paper, we described a specific form of factored MDP (FMDP) which uses a graph-based representation of local dependencies: the GMDP framework. The specificities are: first, each decision node is the parent of exactly one state and no two decision nodes are attached to the same state node. second, the scope of each reward node is included in the set of parents of one state variable and no two reward nodes are attached to the same state variable.

An important feature of GMDP is that they admit problem and approximately optimal solution representations in space linear in the size of the underlying graph. However, classical Dynamic Programming algorithms use space and time exponential in the size of the graph to compute optimal solutions. We have proposed an approximate *Policy Iteration* algorithm which requires only linear space and

time provided that the graph width remains small. The quality loss associated with this gain in time complexity is not known yet. However, experiments showed that it is limited. In particular, the approximate algorithm greatly improves the greedy short-term solution which is usually the only one available in problems of high dimension.

Our approach should be contrasted to [2, 7, 5] which are also devoted to FMDP. [2] uses *algebraic decision diagrams* (ADD) to model and solve FMDP and [7, 5] use a *linear programming* approach. [5] propose a method to tackle FMDP with multiple decision variables based on the idea of sampling the (exponential size) set of constraints generated by a linear programming (LP) modelling of a FMDP. However, to keep acceptable probabilistic bounds on the quality of the returned policy, the sample size should remain proportional to the number of possible actions, i.e. exponential in the number of decision variables. [7] instead consider the exact LP, but propose a specific *variable elimination* method which take the structure of the LP into account in order to solve it exactly. [7] points out experiments in which his method outperforms that of [5] when the same computation time is allowed to both methods.

[2, 7, 5] can only handle problems with a small number of decision variables and are thus inadequate to solve GMDP. In [8], *collaborative multi-agent factored MDP*, which are more general than GMDP, are studied and approximately solved by a dedicated algorithm. The empirical comparison between this algorithm and ours has not been performed yet, but the complexity of this algorithm is quadratic in the number of decision variables, while ours is linear.

Finally, [6] propose a LP-based method for solving GMDP, derived from the method of [7]. Experiments led so far show that even if this approach differs from ours, the returned policies have similar values, and the CPU times are linearly related. We are currently investigating the benefits of each method, in particular their ability to face the following generalisation of GMDP. An important assumption of the GMDP framework is that the action space \mathcal{A} is a cross product of *independent* local action spaces \mathcal{A}_i . However, this assumption should be relaxed (\mathcal{A} becoming a strict subset of $\prod_i \mathcal{A}_i$), as soon as we want to model resource allocation or local constraints between the actions. Relaxing this independence assumption clearly affects the approximation method suggested in this paper and we are currently exploring the effects of this relaxation.

REFERENCES

- [1] E. Altman, *Constrained Markov Decision Processes*, Chapman & Hall / CRC, 1999.
- [2] C. Boutilier, R. Dearden, and M. Goldszmidt, ‘Stochastic dynamic programming with factored representations’, *Artificial Intelligence*, **121**(1), 49–107, (2000).
- [3] D. Chandler, *Introduction to Modern Statistical Mechanics*, Oxford University Press, 1987.
- [4] R. K. Chorom, H. Daduna, and P. S. Knopov, ‘Controlled markov fields with finite state space on graphs’, *Stochastic Models*, (21), 847–874, (2005).
- [5] D. P. de Farias and B. Van Roy, ‘On constraint sampling in the linear programming approach to approximate dynamic programming’, *Math. of Op. Research*, **29**(3), 462–478, (2004).
- [6] N. Forsell and R. Sabbadin, ‘Approximate linear-programming algorithms for gmdp’, in *Proc’ ECAI’06.*, (2006).
- [7] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, ‘Efficient solution algorithms for factored MDPs’, *Journal of Artificial Intelligence Research*, **19**, 399–468, (2003).
- [8] C. E. Guestrin, *Planning under uncertainty in complex structured environments*, Ph.D. dissertation, Stanford University, 2003.
- [9] M. L. Puterman, *Markov Decision Processes*, John Wiley and Sons, New York, 1994.

Unified Definition of Heuristics for Classical Planning

Jussi Rintanen¹

Abstract. In many types of planning algorithms distance heuristics play an important role. Most of the earlier works restrict to STRIPS operators, and their application to a more general language with disjunctivity and conditional effects first requires an exponential size reduction to STRIPS operators.

I present direct formalizations of a number of distance heuristics for a general operator description language in a uniform way, avoiding the exponentiality inherent in earlier reductive approaches. The formalizations use formulae to represent the conditions under which operators have given effects. The exponentiality shows up in satisfiability tests with these formulae, but would appear to be a minor issue because of the small size of the formulae.

1 Introduction

Much of planning research has been carried out in the context of STRIPS operators, named after the famous planning system [3]. A STRIPS operator consists of three sets of facts: a precondition, an *add* list, and a *delete* list. There are many actions that cannot be expressed in terms of STRIPS operators only, for example anything with context-dependent effects, but it is well known that any classical planning problem (one initial state, deterministic actions) can be translated into an equivalent one with STRIPS operators only. However, this translation may increase the size of the planning problem exponentially, and therefore it is not desirable.

In this paper we use a general language for expressing classical planning problems, including conditional effects and disjunctive preconditions. We show that for many purposes a small toolbox of definitions based on the propositional logic is sufficient for handling a general operator definition rather elegantly. Specifically, we consider a number of *distance heuristics* developed for STRIPS planning, and generalize them to a much more expressive planning language.

Anecdotal evidence from the planning community tells that handling conditional effects and disjunctive preconditions elegantly is difficult. Indeed, no clear formalizations of the above heuristics – or many planning algorithms and other planning techniques – in the full generality of conditional effects and disjunctive preconditions have been published. Related works and planning system implementations rely on *ad hoc* techniques, for example on an exponential translation of all formulae into DNF and on eliminating conditional effects. We show how general operator definitions can be handled concisely and elegantly without an exponential blow up in size or processing time.

The outline of this paper is as follows. In Section 4.1 we generalize Bonet and Geffner's [1] max heuristic to operators with conditional effects and arbitrary disjunctive preconditions. A novelty here is that the heuristic is not defined as a cyclic recursion equation. In Section 4.2 we do the same to Bonet and Geffner's [1] additive heuristic, and in Section 4.3 to Hoffmann and Nebel's [7] relaxed plan heuristic.

2 Preliminaries

We give a definition of operators that corresponds to ADL/PDDL operators [4] that have been grounded which means that quantifiers *forall* and *exists* have been eliminated and all parameters have been instantiated with objects of appropriate types in all possible ways.

We use the classical propositional logic with the connectives \vee, \wedge, \neg and the constant symbols *true* \top and *false* \perp . The state variables of a planning problem are the atomic propositions of the logic. State variables $a \in A$ and their negations are *literals*. The *complement* \bar{l} of a literal l is defined by $\bar{a} = \neg a$ and $\bar{\neg a} = a$ for all $a \in A$.

Definition 1. Let A be a set of state variables. An operator over A is a pair $\langle c, e \rangle$ where c is a formula over A describing the precondition and e is an effect over A . Effects are recursively defined as follows.

1. \top is an effect (the dummy effect).
2. a and $\neg a$ for state variables $a \in A$ are effects.
3. $e_1 \wedge \dots \wedge e_n$ is an effect if e_1, \dots, e_n are effects over A (the special case with $n = 0$ is defined as the dummy effect \top .)
4. $c \triangleright e$ is an effect if c is a formula and e is an effect over A .

Conditional effects $c \triangleright e$ mean that e is executed if c is true.

Definition 2 (Operator execution). Let $o = \langle c, e \rangle$ be an operator over A and s a state (an assignment of truth values to A). The operator is executable in s if $s \models c$ and $\{a, \neg a\} \not\subseteq [e]_s$ for all $a \in A$. The set $[e]_s$ of literals (the active effects of e in s) is defined as follows.

1. $[\top]_s = \emptyset$
2. $[a]_s = \{a\}$ for $a \in A$
3. $[\neg a]_s = \{\neg a\}$ for $a \in A$
4. $[e_1 \wedge \dots \wedge e_n]_s = [e_1]_s \cup \dots \cup [e_n]_s$
5. $[c \triangleright e]_s = [e]_s$ if $s \models c$ and $[c \triangleright e]_s = \emptyset$ otherwise.

The successor state $app_o(s)$ of s under o is obtained from s by making the literals in $[e]_s$ true and retaining the values of state variables not occurring in $[e]_s$. For sequences $\sigma = o_1; o_2; \dots; o_n$ of operators we define $app_\sigma(s) = app_{o_n}(\dots app_{o_2}(app_{o_1}(s))\dots)$.

Example 1. Consider the operator $\langle a, e \rangle$ where $e = \neg a \wedge (\neg c \triangleright b)$ and a state s such that $s \models a \wedge b \wedge c$. The operator is executable because $s \models a$. Now $[e]_s = \{\neg a\}$ and $app_{\langle a, e \rangle}(s) \models \neg a \wedge b \wedge c$. ■

A *problem instance* is a quadruple $\langle A, I, O, G \rangle$ where A is the set of state variables, I is the initial state, O is the set of operators and G is the goal formula. A sequence $\sigma = o_1; \dots; o_n$ of operators is a *plan* for the problem instance if $app_\sigma(I) \models G$.

The formula $EPC_I(e)$ expresses the conditions under which the literal l is assigned *true* when the effect e is executed.

¹ National ICT Australia, Canberra Research Laboratory, Canberra, Australia

Definition 3. Define the formula $EPC_l(e)$ recursively as follows.

$$\begin{aligned} EPC_l(\top) &= \perp \\ EPC_l(l) &= \top \\ EPC_l(l') &= \perp \text{ when } l \neq l' \text{ (for literals } l') \\ EPC_l(e_1 \wedge \dots \wedge e_n) &= EPC_l(e_1) \vee \dots \vee EPC_l(e_n) \\ EPC_l(c \triangleright e) &= c \wedge EPC_l(e) \end{aligned}$$

The case $EPC_l(e_1 \wedge \dots \wedge e_n) = EPC_l(e_1) \vee \dots \vee EPC_l(e_n)$ is defined as a disjunction because it is sufficient that at least one of the effects e_1, \dots, e_n makes l true. For example, $EPC_a(a \wedge b) \equiv \top$, $EPC_a(b) \equiv \perp$ and $EPC_a((b \triangleright a) \wedge (c \triangleright a)) \equiv b \wedge c$.

We define a formula that indicates when an operator is executable so that a given literal becomes true.

Definition 4. Let A be the set of state variables and $o = \langle c, e \rangle$ an operator over A . Define

$$EPC_l(o) = c \wedge EPC_l(e) \wedge \bigwedge_{a \in A} \neg(EPC_a(e) \wedge EPC_{\neg a}(e)).$$

There is a connection between $EPC_l(e)$ and the definition of $[e]_s$.

Lemma 1. Let A be the set of state variables, s a state over A , l a literal over A , and o an operator over A with the effect e . Then

1. $l \in [e]_s$ if and only if $s \models EPC_l(e)$, and
2. $app_o(s)$ is defined and $l \in [e]_s$ if and only if $s \models EPC_l(o)$.

The proof of this lemma and some that follow are by structural induction and can be found in a technical report [9].

3 Reachability and Distances

The notion of reachability is important in defining whether a planning problem is solvable and in deriving techniques that speed up search for plans. Distances between states are closely related to reachability and will be defined next. Heuristics in Section 4 are approximations of distances. Define $img_o(S) = \{app_o(s) | s \in S\}$.

Definition 5. Let I be an initial state and O a set of operators. Define the distance sets D_i^{fwd} for I, O which consist of those states that are reachable from I by executing at most i operators.

$$\begin{aligned} D_0^{fwd} &= \{I\} \\ D_i^{fwd} &= D_{i-1}^{fwd} \cup \{s | o \in O, s \in img_o(D_{i-1}^{fwd})\} \text{ for all } i \geq 1 \end{aligned}$$

Definition 6. Given a state I , a set O of operators and the distance sets $D_0^{fwd}, D_1^{fwd}, \dots$ for I, O , the distance of a state s from I is

$$\delta_I^{fwd}(s) = \begin{cases} 0 & \text{if } s = I \\ i & \text{if } s \in D_i^{fwd} \setminus D_{i-1}^{fwd}. \end{cases}$$

If $s \notin D_i^{fwd}$ for all $i \geq 0$ then $\delta_I^{fwd}(s) = \infty$. States that have a finite distance are reachable (from I with O).

Distances can also be defined for formulae.

Definition 7. The distance $\delta_I^{fwd}(\phi)$ of a formula ϕ is $i \geq 1$ if there is a state s such that $s \models \phi$ and $\delta_I^{fwd}(s) = i$ and there is no state s' such that $s' \models \phi$ and $\delta_I^{fwd}(s') < i$. If $I \models \phi$ then $\delta_I^{fwd}(\phi) = 0$.

A formula ϕ has a finite distance iff $\langle A, I, O, \phi \rangle$ has a plan.

Reachability and distances are useful for implementing efficient planning systems. We mention two applications. First, if the precondition of an operator is not reachable, the operator can never be applied and can be ignored. Second, having an oracle that could tell the distances of states for free would directly yield a planning algorithm: step by step choose a sequence of operators that decrease the distance of the goal formula from the current state by one.

Computing distances is in the worst case just as difficult as planning itself (PSPACE-complete), and hence it is usually not useful to use exact distances. Instead, different kinds of *approximations* of distances and reachability can be used.

4 Distance Heuristics

Many approximations of distances are based on the following idea. Instead of considering the number of operators required to reach individual states, we compute approximate numbers of operators to reach states in which certain literals are true. So instead of distances of states, we use distances of literals. Within the computation of distances of literals, distances of formulae are estimated in terms of the distances of literals, which introduces an inaccuracy because the dependencies between literals are ignored. The heuristics therefore may underestimate the actual distance and only yield a lower bound for it.

4.1 Admissible Max Heuristic

Effective heuristics were presented by McDermott [8], Bonet et al. [2, 1] and Haslum and Geffner [6]. We first describe a generalization of the Bonet et al. *max* heuristics to our definition of operators.

We give a procedure that computes a lower bound on the number of operator applications that are needed for reaching from a state I a state in which certain literals are true. Sets D_i^{max} consist of literals that are true in all states that have a distance $\leq i$.

Definition 8. Let $L = A \cup \{\neg a | a \in A\}$ be the set of literals over A and I a state. For all $i \geq 1$ define the max-distance sets

$$\begin{aligned} D_0^{max} &= \{l \in L | I \models l\}, \text{ and} \\ D_i^{max} &= D_{i-1}^{max} \setminus \{l \in L | o \in O, D_{i-1}^{max} \cup \{EPC_l(o)\} \text{ is satisfiable}\}. \end{aligned}$$

The computation starts from D_0^{max} which consists of all literals that are true in the initial state I . This set therefore characterizes those states that have distance 0 from the initial state.

Then we compute sets of literals characterizing states with distance 1, 2 and so on. Each set D_i^{max} is computed from the preceding set D_{i-1}^{max} by testing for each operator o whether it is executable in one of the distance $i - 1$ states and whether it could make a literal l false. This is by testing whether $EPC_l(o)$ is true in one of the distance $i - 1$ states. If it is, l will not be included in D_i^{max} .

Since we consider only finite sets A of state variables and $|D_0^{max}| = |A|$ and $D_{i+1}^{max} \subseteq D_i^{max}$ for all $i \geq 0$, necessarily $D_i^{max} = D_j^{max}$ for some $i \leq |A|$ and all $j > i$.

Every state with distance $\leq i$ satisfies D_i^{max} .

Theorem 9. Let $D_i^{fwd}, i \geq 0$ be the distance sets and D_i^{max} the max-distance sets for I and O . Then for all $i \geq 0$, $D_i^{fwd} \subseteq \{s \in S | s \models D_i^{max}\}$ where S is the set of all states.

The sets D_i^{max} can be used for estimating the distances of formulae. The distance of a formula is the minimum of the distances of states that satisfy it.

Definition 10. Let ϕ be a formula. Define

$$\delta_I^{\max}(\phi) = \begin{cases} 0 & \text{iff } D_0^{\max} \cup \{\phi\} \text{ is satisfiable} \\ d & \text{iff } D_d^{\max} \cup \{\phi\} \text{ is satisfiable and} \\ & D_{d-1}^{\max} \cup \{\phi\} \text{ is not satisfiable, for } d \geq 1. \end{cases}$$

Lemma 2. Let I be a state, O a set of operators, and $D_0^{\max}, D_1^{\max}, \dots$ the max-distance sets for I and O . Then $\text{app}_{o_1, \dots, o_n}(I) \models D_n^{\max}$ for any operators $\{o_1, \dots, o_n\} \subseteq O$.

The next theorem shows that the distance estimates are a lower bound on the actual distances.

Theorem 11. Let I be a state, O a set of operators, ϕ a formula, and $D_0^{\max}, D_1^{\max}, \dots$ the max-distance sets for I and O . If $\text{app}_{o_1, \dots, o_n}(I) \models \phi$, then $D_n^{\max} \cup \{\phi\}$ is satisfiable.

Proof. By Lemma 2 $\text{app}_{o_1, \dots, o_n}(I) \models D_n^{\max}$. By assumption $\text{app}_{o_1, \dots, o_n}(I) \models \phi$. Hence $D_n^{\max} \cup \{\phi\}$ is satisfiable. \square

Corollary 12. For a formula ϕ and a state I , if o_1, \dots, o_n is a sequence of operators and $\text{app}_{o_1, \dots, o_n}(I) \models \phi$ then $n \geq \delta_I^{\max}(\phi)$.

The estimate $\delta_s^{\max}(\phi)$ never overestimates the distance from s to ϕ and it is therefore an admissible heuristic. It may severely underestimate the distance, as discussed in Section 4.2. A family h^m of stronger admissible heuristics has been proposed by Haslum and Geffner [6]. The heuristic h^1 is the max heuristic. There is an algorithm for computing m -literal invariant clauses that generalizes the sets D^{\max} to sets of m -literal clauses [9] and yields the h^m heuristic.

4.1.1 Distance Estimation in Polynomial Time

The computation of the sets D_i^{\max} and the distances $\delta_I^{\max}(\phi)$ is polynomial time except for the NP-complete satisfiability tests for $D \cup \{\phi\}$ where D is a set of literals. Here $\phi = \text{EPC}_I(o)$ for a literal l and an operator o . The size of ϕ is in the worst case quadratic in the size of o , and hence the formulae ϕ are typically small. The cost of the satisfiability test is exponential in the size of ϕ and linear in D , and hence the NP-completeness is almost never an issue.

However, to show that our approach is feasible even in the unrealistic case of huge operators we give a simple polynomial time approximation $\text{asat}(D, \phi)$ of the satisfiability tests with the property that if $D \cup \{\phi\}$ is satisfiable then $\text{asat}(D, \phi) = \text{true}$. The proof of Theorem 9 works when the satisfiability tests of $D \cup \{\phi\}$ are replaced by $\text{asat}(D, \phi)$. The approximation never leads to overestimating the distances, only underestimating, and for STRIPS operators the approximation never makes a difference and for general operators rarely. Define $\text{asat}(D, \phi)$ as follows.

$$\begin{aligned} \text{asat}(D, \perp) &= \text{false} \\ \text{asat}(D, \top) &= \text{true} \\ \text{asat}(D, a) &= \text{true iff } \neg a \notin D \text{ (for } a \in A\text{)} \\ \text{asat}(D, \neg a) &= \text{true iff } a \notin D \text{ (for } a \in A\text{)} \\ \text{asat}(D, \neg\neg\phi) &= \text{asat}(D, \phi) \\ \text{asat}(D, \phi_1 \vee \phi_2) &= \text{asat}(D, \phi_1) \text{ or } \text{asat}(D, \phi_2) \\ \text{asat}(D, \phi_1 \wedge \phi_2) &= \text{asat}(D, \phi_1) \text{ and } \text{asat}(D, \phi_2) \\ \text{asat}(D, \neg(\phi_1 \vee \phi_2)) &= \text{asat}(D, \neg\phi_1) \text{ and } \text{asat}(D, \neg\phi_2) \\ \text{asat}(D, \neg(\phi_1 \wedge \phi_2)) &= \text{asat}(D, \neg\phi_1) \text{ or } \text{asat}(D, \neg\phi_2) \end{aligned}$$

The cases for $\neg(\phi_1 \wedge \phi_2)$ and $\neg(\phi_1 \vee \phi_2)$ are respectively obtained from the cases for $\phi_1 \vee \phi_2$ and $\phi_1 \wedge \phi_2$ by the De Morgan laws.

The inaccuracy of the satisfiability tests stems from the fact that for formulae $\phi_1 \wedge \phi_2$ (respectively $\neg(\phi_1 \vee \phi_2)$) we make recursively

two satisfiability tests that do not require that ϕ_1 and ϕ_2 (respectively $\neg\phi_1$ and $\neg\phi_2$) are simultaneously satisfiable. For example, $\text{asat}(\emptyset, a \wedge \neg a) = \text{true}$ although $a \wedge \neg a$ is unsatisfiable.

Lemma 3. Let ϕ be a formula and D a set of literals. If $D \cup \{\phi\}$ is satisfiable, then $\text{asat}(D, \phi)$ returns true.

4.2 Inadmissible Additive Heuristic

The max heuristic is very optimistic about the distances, and in many cases it seriously underestimates them. If there are two goal literals, the maximum of the goal costs (distances) is taken to be the combined cost. This however is only accurate when the easier goal is achieved for free while achieving the more difficult one. Goals are often independent and then a more accurate estimate would be the sum of the individual costs. To fix this problem Bonet and Geffner [1] proposed a more practical variant of the max heuristic.

Our definition is based on an auxiliary definition for the cost of achieving a formula. The cost of a literal is the number of steps it takes to achieve it, in a similar approximative sense as in the max heuristic. The cost of a disjunction is the minimum cost of the disjuncts. The cost of a conjunction is the sum of the costs of the conjuncts. This is the difference to the max heuristic where the cost of a conjunction is the maximum of the costs of the conjuncts.

Definition 13. Let I be a state and $L = A \cup \{\neg a | a \in A\}$ the set of literals. Define the sets D_i^+ for $i \geq 0$ as follows.

$$\begin{aligned} D_0^+ &= \{l \in L | I \models l\} \\ D_i^+ &= D_{i-1}^+ \setminus \{l \in L | o \in O, \text{cost}(EP\bar{C}_I(o), i) < i\} \text{ for all } i \geq 1 \end{aligned}$$

We define $\text{cost}(\phi, i)$ as follows.

$$\begin{aligned} \text{cost}(\perp, i) &= \infty \\ \text{cost}(\top, i) &= 0 \\ \text{cost}(a, i) &= 0 \text{ if } \neg a \notin D_0^+, \text{ for } a \in A \\ \text{cost}(\neg a, i) &= 0 \text{ if } a \notin D_0^+, \text{ for } a \in A \\ \text{cost}(a, i) &= j \text{ if } \neg a \in D_{j-1}^+ \setminus D_j^+ \text{ for some } j < i \\ \text{cost}(\neg a, i) &= j \text{ if } a \in D_{j-1}^+ \setminus D_j^+ \text{ for some } j < i \\ \text{cost}(a, i) &= \infty \text{ if } \neg a \in D_j^+ \text{ for all } j < i \\ \text{cost}(\neg a, i) &= \infty \text{ if } a \in D_j^+ \text{ for all } j < i \\ \text{cost}(\phi_1 \vee \phi_2, i) &= \min(\text{cost}(\phi_1, i), \text{cost}(\phi_2, i)) \\ \text{cost}(\phi_1 \wedge \phi_2, i) &= \text{cost}(\phi_1, i) + \text{cost}(\phi_2, i) \\ \text{cost}(\neg\phi, i) &= \text{cost}(\phi, i) \\ \text{cost}(\neg(\phi_1 \wedge \phi_2), i) &= \min(\text{cost}(\neg\phi_1, i), \text{cost}(\neg\phi_2, i)) \\ \text{cost}(\neg(\phi_1 \vee \phi_2), i) &= \text{cost}(\neg\phi_1, i) + \text{cost}(\neg\phi_2, i) \end{aligned}$$

The above definitions would appear to be cyclic but the definition of D_i^+ refers to $\text{cost}(\phi, j)$ for $j \leq i$ only and the definition of $\text{cost}(\phi, i)$ refers to D_j^+ for $j < i$ only.

The definition of $\text{cost}(\phi, i)$ ignores dependencies between conjuncts in the same way as the definition $\text{asat}(D, \phi)$.

Definition 14. Let ϕ be a formula. Define

$$\delta_I^+(\phi) = \text{cost}(\phi, n)$$

where n is the smallest i such that $D_i^+ = D_{i-1}^+$.

The following theorem shows that the distance estimates of the additive heuristic are never lower than those of the max heuristic.

Theorem 15. Let $D_i^+, i \geq 0$ be the sets defined in terms of the approximate tests $\text{asat}(D, \phi)$. Then $D_i^{\max} \subseteq D_i^+$ for all $i \geq 0$.

Proof. The proof is by induction on i .

Base case $i = 0$: By definition $D_0^+ = D_0^{max}$.

Inductive case $i \geq 1$: By the induction hypothesis $D_{i-1}^{max} \subseteq D_{i-1}^+$. To show $D_{i-1}^{max} \setminus \{l \in L \mid o \in O, \text{asat}(D_{i-1}^{max}, EPC_{\bar{l}}(o))\} \subseteq D_{i-1}^+ \setminus \{l \in L \mid o \in O, \text{cost}(EPC_{\bar{l}}(o), i) < i\}$ it is sufficient to show that $\text{cost}(EPC_{\bar{l}}(o), i) < i$ implies $\text{asat}(D_{i-1}^{max}, EPC_{\bar{l}}(o))$.

We show this by induction on the structure of $\phi = EPC_{\bar{l}}(o)$.

Induction hypothesis: $\text{cost}(\phi, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi) = \text{true}$.

Base case 1, $\phi = \perp$: $\text{cost}(\perp, i) = \infty$ and $\text{asat}(D_i^{max}, \perp) = \text{false}$.

Base case 2, $\phi = \top$: $\text{cost}(\top, i) = 0$ and $\text{asat}(D_i^{max}, \top) = \text{true}$.

Base case 3, $\phi = a$: If $\text{cost}(a, i) < i$ then $\neg a \notin D_j^+$ for some $j < i$ or $\neg a \notin D_0^+$. Hence $\neg a \notin D_{i-1}^+$. By the outer induction hypothesis $\neg a \notin D_{i-1}^{max}$ and consequently $\neg a \notin D_i^{max}$. Hence $\text{asat}(D_i^{max}, a) = \text{true}$.

Base case 4, $\phi = \neg a$: Analogous to the case $\phi = a$.

Inductive case 5, $\phi = \phi_1 \vee \phi_2$: Assume $\text{cost}(\phi_1 \vee \phi_2, i) < i$. Since $\text{cost}(\phi_1 \vee \phi_2, i) = \min(\text{cost}(\phi_1, i), \text{cost}(\phi_2, i))$, either $\text{cost}(\phi_1, i) < i$ or $\text{cost}(\phi_2, i) < i$. By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$, and $\text{cost}(\phi_2, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_2)$. Hence either $\text{asat}(D_{i-1}^{max}, \phi_1)$ or $\text{asat}(D_{i-1}^{max}, \phi_2)$. Therefore by definition $\text{asat}(D_{i-1}^{max}, \phi_1 \vee \phi_2)$.

Inductive case 6, $\phi = \phi_1 \wedge \phi_2$: Assume $\text{cost}(\phi_1 \wedge \phi_2, i) < i$. Since $i \geq 1$ and $\text{cost}(\phi_1 \wedge \phi_2, i) = \text{cost}(\phi_1, i) + \text{cost}(\phi_2, i)$, both $\text{cost}(\phi_1, i) < i$ and $\text{cost}(\phi_2, i) < i$. By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$, and $\text{cost}(\phi_2, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_2)$. Hence both $\text{asat}(D_{i-1}^{max}, \phi_1)$ and $\text{asat}(D_{i-1}^{max}, \phi_2)$. Therefore by definition $\text{asat}(D_{i-1}^{max}, \phi_1 \wedge \phi_2)$.

Inductive case 7, $\phi = \neg\neg\phi_1$: By the induction hypothesis $\text{cost}(\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \phi_1)$. By definition $\text{cost}(\neg\neg\phi_1, i) = \text{cost}(\phi_1, i)$ and $\text{asat}(D, \neg\neg\phi) = \text{asat}(D, \phi)$. By the induction hypothesis $\text{cost}(\neg\neg\phi_1, i) < i$ implies $\text{asat}(D_{i-1}^{max}, \neg\neg\phi_1)$.

Inductive case 8, $\phi = \neg(\phi_1 \vee \phi_2)$: Analogously to $\phi = \phi_1 \wedge \phi_2$.

Inductive case 9, $\phi = \neg(\phi_1 \wedge \phi_2)$: Analogously to $\phi = \phi_1 \vee \phi_2$. \square

That the additive heuristic gives higher estimates than the max heuristic could in many cases be viewed as an advantage because the estimates would be more accurate. However, sometimes this leads to overestimating the actual distance, and the heuristic is therefore not admissible (but see recent work by Haslum et al. [5].)

Example 2. Consider an initial state such that $I \models \neg a \wedge \neg b \wedge \neg c$ and the operator $\langle \top, a \wedge b \wedge c \rangle$. A state satisfying $a \wedge b \wedge c$ is reached by this operator in one step but $\delta_I^+(a \wedge b \wedge c) = 3$. \blacksquare

4.3 Inadmissible Relaxed Plan Heuristic

The max heuristic and the additive heuristic represent two extremes. The first assumes that sets of operators required for reaching the different goal literals maximally overlap in the sense that the operators for the most difficult goal literal include the operators for all the remaining ones. The second assumes that these sets are completely disjoint. In some cases the first is better and in others the second.

To estimate the distances more accurately the operators should be better taken into account, which suggests yet another approach to computing a heuristic: attempt to find a set of operators that in a very loose sense reaches the goals. This idea has been considered by Hoffmann and Nebel [7]. A *relaxed plan* is computed as follows (see the algorithm in Figure 1.) We first choose a set of goal literals the truth of which is sufficient for the truth of the goal formula G . These literals must be reachable in the sense of the sets D_i^{max} from Section 4.1. Then we identify the goal literals l with the highest distance

(in the D_i^{max} sense) and a set of operators making them true. These operators form the last step of the relaxed plan. A new goal formula represents the conditions under which these operators can make the literals true. Then a new set of goal literals is produced by a form of regression from the new goal formula. The computation is repeated until we have a set of goal literals that are true in the initial state. Along the way we have constructed a relaxed plan.

The function $\text{goals}(D, \phi)$ recursively finds a set M of literals such that $M \models \phi$ and each literal in M is consistent with D . Note that M itself is not necessarily consistent, for example for $D = \emptyset$ and $\phi = a \wedge \neg a$ we get $M = \{a, \neg a\}$. If a set M is found $\text{goals}(D, \phi) = \{M\}$ and otherwise $\text{goals}(D, \phi) = \emptyset$.

Definition 16. Let D be a set of literals.

$$\begin{aligned} \text{goals}(D, \perp) &= \emptyset \\ \text{goals}(D, \top) &= \{\emptyset\} \\ \text{goals}(D, a) &= \{\{a\}\} \text{ if } \neg a \notin D \\ \text{goals}(D, a) &= \emptyset \text{ if } \neg a \in D \\ \text{goals}(D, \neg a) &= \{\{\neg a\}\} \text{ if } a \notin D \\ \text{goals}(D, \neg a) &= \emptyset \text{ if } a \in D \\ \text{goals}(D, \neg\neg\phi) &= \text{goals}(D, \phi) \\ \text{goals}(D, \phi_1 \vee \phi_2) &= \begin{cases} \text{goals}(D, \phi_1) \text{ if } \text{goals}(D, \phi_1) \neq \emptyset \\ \text{goals}(D, \phi_2) \text{ otherwise} \end{cases} \\ \text{goals}(D, \phi_1 \wedge \phi_2) &= \begin{cases} \{L_1 \cup L_2\} \text{ if } \text{goals}(D, \phi_1) = \{L_1\} \\ \quad \text{and } \text{goals}(D, \phi_2) = \{L_2\} \\ \emptyset \quad \text{otherwise} \end{cases} \\ \text{goals}(D, \neg(\phi_1 \wedge \phi_2)) &= \begin{cases} \text{goals}(D, \neg\phi_1) \text{ if } \text{goals}(D, \neg\phi_1) \neq \emptyset \\ \text{goals}(D, \neg\phi_2) \text{ otherwise} \end{cases} \\ \text{goals}(D, \neg(\phi_1 \vee \phi_2)) &= \begin{cases} \{L_1 \cup L_2\} \text{ if } \text{goals}(D, \neg\phi_1) = \{L_1\} \\ \quad \text{and } \text{goals}(D, \neg\phi_2) = \{L_2\} \\ \emptyset \quad \text{otherwise} \end{cases} \end{aligned}$$

If both ϕ_1 and ϕ_2 yield goal literals for $\phi_1 \vee \phi_2$ the set for ϕ_1 is chosen. A practically better choice is the smaller of the two sets.

Lemma 4. Let D be a set of literals and ϕ a formula.

1. $\text{goals}(D, \phi) \neq \emptyset$ if and only if $\text{asat}(D, \phi) = \text{true}$.
2. If $\text{goals}(D, \phi) = \{M\}$ then $\{\bar{l} \mid l \in M\} \cap D = \emptyset$ and $\text{asat}(D, \bigwedge_{l \in M} l) = \text{true}$.

Proof. 1. This is an easy induction proof on the structure of ϕ based on the definitions of $\text{asat}(D, \phi)$ and $\text{goals}(D, \phi)$.

2. This is because $\bar{l} \notin D$ for all $l \in M$. This can be shown by a simple induction proof. \square

Lemma 5. Let D and $D' \subseteq D$ be sets of literals. If $\text{goals}(D, \phi) = \emptyset$ and $\text{goals}(D', \phi) = \{M\}$, then there is $l \in M$ such that $\bar{l} \in D \setminus D'$.

Definition 17. Define $\delta_I^{rlx}(\phi) = \text{relaxedplan}(A, I, O, \phi)$.

The additive and the relaxed plan heuristic are incomparable and both give estimates at least as high as the max heuristic.

Theorem 18. Let ϕ be a formula and $\delta_I^{max}(\phi)$ the max-distance defined in terms of $\text{asat}(D, \phi)$. Then $\delta_I^{rlx}(\phi) \geq \delta_I^{max}(\phi)$.

Proof. We have to show that for any formula G the procedure call $\text{relaxedplan}(A, I, O, G)$ returns a number $\geq \delta_I^{max}(G)$.

The procedure returns ∞ if and only if $\text{asat}(D_0^{max}, G) = \text{false}$ for all $i \geq 0$. In this case by definition $\delta_I^{max}(G) = \infty$. Otherwise $t = \delta_I^{max}(G)$. Now $t = 0$ if and only if $\text{asat}(D_0^{max}, G) = \text{true}$. In this case the procedure returns 0 without iterating the loop (line 9.)

```

1: procedure relaxedplan(A,I,O,G);
2:    $L := A \cup \{\neg a | a \in A\}$ ; (* Set of all literals *)
3:   compute sets  $D_i^{\max}$  as in Definition 8;
4:   if asat( $D_i^{\max}, G$ ) = false for all  $i \geq 0$  then return  $\infty$ ;
5:    $t := \delta_I^{\max}(G)$ ;
6:    $L_{t+1}^G := \emptyset$ ; (* Goal literals initially *)
7:    $N_{t+1} := \emptyset$ ;
8:    $G_t := G$ ; (* Initial goal formula *)
9:   for  $i := t$  downto 1 do
10:     $L_i^G := (L_{i+1}^G \setminus N_{i+1}) \cup \{l \in M | M \in \text{goals}(D_i^{\max}, G_i)\}$ ;
11:     $N_i := \{l \in L_i^G | l \in D_{i-1}^{\max}\}$ ; (* Goals becoming true at i *)
12:     $T_i := \text{a minimal subset of } O$  (* Operators making  $N_i$  true *)
13:    so that  $N_i \subseteq \{l \in L_i | o \in T_i, \text{asat}(D_{i-1}^{\max}, EPC_l(o))\}$ ;
14:     $G_{i-1} := \bigwedge_{l \in N_i} \bigvee \{EPC_l(o) | o \in T_i\}$ ; (* New goal *)
15:   end do
16:   return  $|T_1| + |T_2| + \dots + |T_t|$ ;

```

Figure 1. Algorithm for finding a relaxed plan

We show that if $t \geq 1$ then $T_i \neq \emptyset$ for every $i \in \{1, \dots, t\}$, entailing $|T_1| + \dots + |T_t| \geq t = \delta_I^{\max}(G)$. This is by induction from t to 1. We use the following auxiliary result. If $\text{asat}(D_{i-1}^{\max}, G_i) = \text{false}$ and $\text{asat}(D_i^{\max}, G_i) = \text{true}$ and $\bar{l} \notin D_i^{\max}$ for all $l \in L_i^G$ then T_i is well-defined and $T_i \neq \emptyset$. The proof is as follows.

By Lemma 4 $\text{goals}(D_{i-1}^{\max}, G_i) = \emptyset$ and $\text{goals}(D_i^{\max}, G_i) = \{M\}$ for some M . By Lemma 5 there is $l \in M$ such that $\bar{l} \in D_{i-1}^{\max}$ and hence $N_i \neq \emptyset$. By definition $\bar{l} \in D_{i-1}^{\max}$ for all $l \in N_i$. By $N_i \subseteq L_i^G$ and the assumption about L_i^G $\bar{l} \notin D_i^{\max}$ for all $l \in N_i$. Hence $\bar{l} \in D_{i-1}^{\max} \setminus D_i^{\max}$ for all $l \in N_i$. Hence by definition of D_i^{\max} for every $l \in N_i$ there is $o \in O$ such that $\text{asat}(D_{i-1}^{\max}, EPC_l(o))$. Hence there is $T_i \subseteq O$ so that $N_i \subseteq \{l \in L_i | o \in T_i, \text{asat}(D_{i-1}^{\max}, EPC_l(o))\}$ and the value of T_i is defined. As $N_i \neq \emptyset$ also $T_i \neq \emptyset$.

In the induction proof we establish the assumptions of the auxiliary result and then invoke the auxiliary result itself.

Induction hypothesis: For all $j \in \{i, \dots, t\}$

1. $\bar{l} \notin D_j^{\max}$ for all $l \in L_j^G$,
2. $\text{asat}(D_j^{\max}, G_j) = \text{true}$ and $\text{asat}(D_{j-1}^{\max}, G_j) = \text{false}$, and
3. $T_j \neq \emptyset$.

Base case $i = t$:

1. $\bar{l} \notin D_t^{\max}$ for all $l \in L_t^G$ by (2) of Lemma 4 because $L_t^G = \{l \in \text{goals}(D_t^{\max}, G_t)\}$.
2. As $t = \delta_I^{\max}(G_t)$ by definition $\text{asat}(D_{t-1}^{\max}, G_t) = \text{false}$ and $\text{asat}(D_t^{\max}, G_t) = \text{true}$.
3. By the auxiliary result from the preceding case.

Inductive case $i < t$:

1. We have $\bar{l} \notin D_i^{\max}$ for all $l \in L_i^G$ because $L_i^G = (L_{i+1}^G \setminus N_{i+1}) \cup \{l \in \text{goals}(D_i^{\max}, G_i)\}$ and by the induction hypothesis $\bar{l} \notin D_{i+1}^{\max}$ for all $l \in L_{i+1}^G$ and by (2) of Lemma 4 $\bar{l} \notin D_i^{\max}$ for all $l \in M$ for $M \in \text{goals}(D_i^{\max}, G_i)$.
2. By definition $G_i = \bigwedge_{l \in N_{i+1}} \bigvee \{EPC_l(o) | o \in T_{i+1}\}$. By definition of T_{i+1} for every $l \in N_{i+1}$ there is $o \in T_{i+1}$ such that $\text{asat}(D_i^{\max}, EPC_l(o)) = \text{true}$. By definition of $\text{asat}(D_i^{\max}, \phi_1 \vee \phi_2)$ and $\text{asat}(D_i^{\max}, \phi_1 \wedge \phi_2)$ also $\text{asat}(D_i^{\max}, G_i) = \text{true}$. Then we show that $\text{asat}(D_{i-1}^{\max}, G_i) = \text{false}$. By definition of D_i^{\max} , $\text{asat}(D_{i-1}^{\max}, EPC_{\bar{l}}(o)) = \text{false}$ for all $l \in D_i^{\max}$ and $o \in O$. Hence $\text{asat}(D_{i-1}^{\max}, EPC_l(o)) = \text{false}$ for all $l \in N_{i+1}$ and $o \in O$.

because $\bar{l} \in D_i^{\max}$. Hence $\text{asat}(D_{i-1}^{\max}, EPC_l(o)) = \text{false}$ for all $l \in N_{i+1}$ and $o \in T_{i+1}$ because $T_{i+1} \subseteq O$. By definition $G_i = \bigwedge_{l \in N_{i+1}} \bigvee \{EPC_l(o) | o \in T_{i+1}\}$. Hence by definition of $\text{asat}(D, \phi)$ also $\text{asat}(D_{i-1}^{\max}, G_i) = \text{false}$.

3. By the auxiliary result from the preceding case. \square

5 Conclusions

We have shown how a number of distance heuristics can be elegantly generalized to an expressive planning language with conditional effects and arbitrary formulae. In comparison to the reductive approach for handling general operators, our approach does not suffer from the problem of exponential size of operator sets. The only exponentiality is in the satisfiability tests, but as the structure of formulae $EPC_l(o)$ is usually very simple even in cases that are difficult to the reductive approach, the worst-case exponentiality is not a serious issue, and can be completely avoided by approximate satisfiability tests.

We have also clarified relations between different heuristics. Hoffmann and Nebel [7] do not elaborate on the close connection between their Graphplan-based relaxed plan heuristic and Bonet and Geffner's max heuristic. Our formalization of the former makes the connection very clear: a relaxed plan is a set of operator occurrences that in a certain sense covers the changes between sets D_i^{\max} and D_{i+1}^{\max} that are relevant for reaching the goals. Planning graphs are not needed for defining the relaxed plan heuristic.

Acknowledgements

This research was supported by National ICT Australia (NICTA), in connection with the DPOLP project. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

REFERENCES

- [1] Blai Bonet and Héctor Geffner, 'Planning as heuristic search', *Artificial Intelligence*, **129**(1-2), 5–33, (2001).
- [2] Blai Bonet, Gábor Loerincs, and Héctor Geffner, 'A robust and fast action selection mechanism for planning', in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, pp. 714–719, Menlo Park, California, (July 1997). AAAI Press.
- [3] Richard E. Fikes and Nils J. Nilsson, 'STRIPS: a new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, **2**(2-3), 189–208, (1971).
- [4] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, 'PDDL - the Planning Domain Definition Language, version 1.2', Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, (October 1998).
- [5] Patrik Haslum, Blai Bonet, and Héctor Geffner, 'New admissible heuristics for domain-independent planning', in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, pp. 1163–1168, (2005).
- [6] Patrik Haslum and Héctor Geffner, 'Admissible heuristics for optimal planning', in *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, eds., Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, pp. 140–149. AAAI Press, (2000).
- [7] J. Hoffmann and B. Nebel, 'The FF planning system: Fast plan generation through heuristic search', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).
- [8] Drew V. McDermott, 'Using regression-match graphs to control search in planning', *Artificial Intelligence*, **109**(1-2), 111–159, (1999).
- [9] Jussi Rintanen, 'State-space traversal techniques for planning', Report 220, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, (2005).

8. PAIS

This page intentionally left blank

Applying Trip@dvice Recommendation Technology to www.visiteurope.com

Adriano Venturini¹ and Francesco Ricci²

Abstract. Implementing decision support technologies in a real commercial tourism destination portal is challenging. First of all, the peculiar problems related to the tourism domain, which have been studied in the recent years in eCommerce and tourism research, must be considered. But, to provide an effective and useful tool, one must tackle additional requirements arising from the technical and operational environment, which influence not only the software development and architectural issues, but also methodological aspects. This paper describes the main choices we taken and the approach we followed to integrate the Trip@dvice recommendation technology in www.visiteurope.com, the major European tourism portal launched in March 2006, with the goal of promoting Europe as a tourist destination.

1 INTRODUCTION

There is a growing number of web sites that support a traveller in the selection of travel destinations or travel products and services (e.g., events, attractions, flight, hotel). Travellers enter tourism portals searching for information about destinations and related topics, such as, points of interests, historical data, weather conditions, travel packages, flights and hotels. This wide spectrum of information is currently provided by a number of web actors belonging to different categories: on-line travel agencies; tour operators; cruise operators; destination management organizations (multi-destination, regional, city); airlines; hotel chains; convention and visitors bureau [2, 15].

To help visitors to find the information they need and plan their vacation, decision aid technologies are being adopted by tourism portals. These tools must take into account the complexity of the tourist decision process, that has already been described in several studies (see [5] for a survey). A destination is a complex concept, it is an aggregation of more elementary components, such as attractions or cultural themes, more than a geographic area. In addition, a "travel plan", i.e., the output of the decision process, may vary greatly in the structure and content, it could be a flight and hotel, or a complete day by day schedule of the holiday. Because of this, the straightforward implementation of general decision aid and recommendation technologies, already proved to be successful in other sectors [12, 1], cannot be successfully applied to travel planning and destination choice [10].

To overcome these limitations, we have developed Trip@dvice, a travel recommendation methodology that incorporates a human choice model derived from specialized literature on traveler's behavior [4], and extends recommender systems designed for simpler products, and in particular those based on Case Base Reasoning [6].

Trip@dvice supports the selection of travel products (e.g., a hotel or a visit to a museum or a climbing school) and the building of a *travel plan*. In our approach the case base is composed of travel plans built by a community of users. A case is structured hierarchically including components that represent the search/decision problem definition, i.e. the travel's and travellers' characteristics, and the problem solution, that is, the set of products and services included in the plan [11].

Trip@dvice has been chosen by the European Union and by the European Travel Commission (ETC) to provide personalized travel recommendations in the European tourism destination portal, www.visiteurope.com. The main goal of [visiteurope.com](http://www.visiteurope.com), which has been launched on March 2006, is to promote Europe as a tourist destination. The Travel Planner tool, based on Trip@dvice technology, is an advanced service allowing tourists to self-bundle their travel plan by receiving personalized recommendations about interesting places and activities that can be practiced in Europe.

To integrate a recommendation technology such as Trip@dvice in a real portal we faced several challenges, normally not considered when dealing with research tools. A tourism destination portal is a complex system composed of a wide set of components and services designed to help the user to find, examine and select the tourist products she is looking for. Supporting the portal visitors in their decisional process means to integrate the recommendation process in the portal concept itself, sharing content, functions and tools of the overall portal. Visitors should be able to bundle their travel plan at any time in their preferred way, either browsing the content through the classical navigation functions, or by selecting the preferred tourist products shown on the home page, or by accepting a suggestion which has been sent via email by a friend.

Therefore the recommender system should exploit the same content repository available in the portal. The European tourism destination portal, stores content related to the 34 European official national tourism organization belonging to ETC, and supports a weakly structured content model representing a wide sets of concepts. Recommendation functions must be adapted to deal with items represented with semi-structured models annotated with metadata information. Furthermore, the tourist target market of the portal is the whole world, thus the portal has to support a wide spectrum of visitors with different characteristics and decision styles [16]. Most of [visiteurope.com](http://www.visiteurope.com) visitors do not know much about Europe and they are supposed to increase their "destination" knowledge with the portal support.

This paper is organized as follows: Section 2 describes the functions implemented by Travel Planner service and the typical interaction supported by the tool. Section 3 summarizes the methodology implemented by Trip@dvice and how it has been adapted to [visiteurope.com](http://www.visiteurope.com).

¹ eCTRL Solutions, Italy, email: venturini@ectrlsolutions.com

² ITC-irst, Italy, email: ricci@itc.it

rope.com. Section 4 gives an overview of the architecture we adopted to integrate the recommendation technologies within the portal. Section 5 deals with some open questions and next research activities.

2 THE TRAVEL PLANNER SERVICE AND SUPPORTED SCENARIO

The Travel Planner service provides a set of functions that allow users to plan their vacation in Europe. It manages the current user travel plan, which is a set of tourist items (e.g. regions, cities, attractions, activities, events) selected by the user to self-bundle the travel. Tourist items can be selected from any section of the portal and added to the user's personal travel plan: each tourist item shown by any information service of the portal can be selected and added to the user's personal travel plan. All the items that the user has added to his personal travel plan contribute to create his user model (of the current travel) and will be exploited by the recommendation functions to provide relevant recommendations (see Section 3).

Figure 1. Specifying Travel Preferences

The initial step of the recommendation process consists in collecting the user's travel preferences, as shown in Figure 1. Here the user can enter general needs and preferences about the travel he is going to plan. Users are not forced to answer all these questions to get recommendations or to use the travel planner services, since this is not the unique source of information for recommendations. Moreover, users can at anytime refine their preferences to better specify their profile. It is important to highlight that the features that the user can here specify (e.g. themes and regions of interests) are not predefined by the recommender system. In fact, they belong to the classification trees used to classify items in the content management system of the portal. Features can be added and removed by the portal content manager, adapting the tool to the evolving needs of the tourism organization. Travel preferences become part of the current case and will be exploited by the Search Activity and by the Seeking for Inspiration functions to deliver product recommendations.

In the Search Activity function the user further specifies detailed preferences about the activities sought (Figure 2). Additional criteria can also be set, like the preferred countries and themes of interests. Note that the top level categories (e.g. Nature) already selected in the Travel Preference section are automatically expanded to allow the user to select children categories (e.g. Natural Park, Landscape) and refine her interests definition. By pushing the search button, activities matching the detailed preferences are ranked and shown to the user, who can add the preferred ones to his travel plan (Figure 3).

Search for an interesting activity

Topic of Interest : <input type="checkbox"/> Sport <input checked="" type="checkbox"/> Nature <input type="checkbox"/> National park <input type="checkbox"/> Landscapes <input type="checkbox"/> Health <input type="checkbox"/> Adventure <input type="checkbox"/> Exhibitions <input type="checkbox"/> Activities <input type="checkbox"/> Services <input type="checkbox"/> Art <input type="checkbox"/> Culture <input checked="" type="checkbox"/> Religion <input type="checkbox"/> History <input type="checkbox"/> Tradition <input type="checkbox"/> Special Interest <input type="checkbox"/> Industrial heritage Type of destination : <input type="checkbox"/> Type of destination	Do you want to change your Travel Plan Preferences ?																																																																														
Region : <table border="0"> <tr> <td><input type="checkbox"/> Central Europe</td> <td><input type="checkbox"/> Alps</td> <td><input type="checkbox"/> Scandinavia</td> <td><input type="checkbox"/> Denmark</td> <td><input type="checkbox"/> Finland</td> <td><input type="checkbox"/> Iceland</td> </tr> <tr> <td><input checked="" type="checkbox"/> Scandinavia</td> <td></td> <td></td> <td><input type="checkbox"/> Norway</td> <td><input type="checkbox"/> Sweden</td> <td></td> </tr> <tr> <td><input type="checkbox"/> Benelux</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Eastern Europe</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Mediterranean</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Iberian peninsula</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> UK and Ireland</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> The Pyrenees</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Northern Europe</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Southern Europe</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Baltic states</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Black Sea</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> Western Europe</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>		<input type="checkbox"/> Central Europe	<input type="checkbox"/> Alps	<input type="checkbox"/> Scandinavia	<input type="checkbox"/> Denmark	<input type="checkbox"/> Finland	<input type="checkbox"/> Iceland	<input checked="" type="checkbox"/> Scandinavia			<input type="checkbox"/> Norway	<input type="checkbox"/> Sweden		<input type="checkbox"/> Benelux						<input type="checkbox"/> Eastern Europe						<input type="checkbox"/> Mediterranean						<input type="checkbox"/> Iberian peninsula						<input type="checkbox"/> UK and Ireland						<input type="checkbox"/> The Pyrenees						<input type="checkbox"/> Northern Europe						<input type="checkbox"/> Southern Europe						<input type="checkbox"/> Baltic states						<input type="checkbox"/> Black Sea						<input type="checkbox"/> Western Europe					
<input type="checkbox"/> Central Europe	<input type="checkbox"/> Alps	<input type="checkbox"/> Scandinavia	<input type="checkbox"/> Denmark	<input type="checkbox"/> Finland	<input type="checkbox"/> Iceland																																																																										
<input checked="" type="checkbox"/> Scandinavia			<input type="checkbox"/> Norway	<input type="checkbox"/> Sweden																																																																											
<input type="checkbox"/> Benelux																																																																															
<input type="checkbox"/> Eastern Europe																																																																															
<input type="checkbox"/> Mediterranean																																																																															
<input type="checkbox"/> Iberian peninsula																																																																															
<input type="checkbox"/> UK and Ireland																																																																															
<input type="checkbox"/> The Pyrenees																																																																															
<input type="checkbox"/> Northern Europe																																																																															
<input type="checkbox"/> Southern Europe																																																																															
<input type="checkbox"/> Baltic states																																																																															
<input type="checkbox"/> Black Sea																																																																															
<input type="checkbox"/> Western Europe																																																																															
Type of recommendation: <input type="radio"/> Activity <input type="radio"/> Route <input type="radio"/> Place to visit																																																																															
<input type="button" value="Search →"/> <input type="button" value="Reset →"/>																																																																															

Figure 2. Specifying Detailed Preferences

The recommendation functions we have implemented in [visit-europe.com](http://www.visit-europe.com) are the followings:

- **Search Activity.** Exploiting the information and preferences acquired during the user-system interaction and the experiences contained in past recommendation sessions, the system identifies and recommends ranked tourist items.
- **Seeking for Inspiration.** This function is intended for those users that rather than specifying preferences and constraints, prefer to browse the portal content, looking and rating proposals made by the system. Each system proposal is a coherent bundle of tourist items. If the user is interested in one proposal, she can either add all or some of its components to the travel plan, or get additional proposals similar to the one that she likes.

The other main recommendation function which has been implemented is called "Seeking for Inspiration" (Figure 4). Here the system initially proposes some alternative bundles of tourist items. The user can examine each proposal in details and add all or part of it to her travel plan, or can get additional recommendations by selecting one bundle ("My favorite is this"). In this case, the system, exploiting the whole history of alternative bundles preferred by the user, proposes new bundles that are supposed to be closer to the user ideal travel.

Results of your search

The screenshot shows a search results page with three items:

- Glacier walking**: Rating 5 stars. Description: "Norway still shows traces of the Ice Age, when the entire country was covered by ice." Buttons: "Suggested experiences" and "add to your Travel Plan".
- Reindeer sledding**: Rating 5 stars. Description: "Reindeer sledding is a unique way of experiencing the Norwegian countryside." Buttons: "Suggested experiences" and "add to your Travel Plan".
- Nature trails**: Rating 5 stars. Description: "If you like learning about the features of the route while you are walking, a nature trail is an excellent option." Buttons: "Suggested experiences" and "add to your Travel Plan".

Figure 3. Search Results

Seeking Inspirations

The screenshot shows a section titled "Seeking Inspirations" with six alternatives:

- ALTERNATIVE 1**: Switzerland Food, Wines. Description: "Gastronomy & Wine in Switzerland". Buttons: "detail" and "My favourite is this!".
- ALTERNATIVE 2**: Switzerland, Germany Food, Wines. Description: "Gastronomy & Wine in Switzerland". Buttons: "detail" and "My favourite is this!".
- ALTERNATIVE 3**: Mediterranean, Cyprus Sightseeing, Culture. Description: "Beaches". Buttons: "detail" and "My favourite is this!".
- ALTERNATIVE 4**: Switzerland Golfing, Skating. Description: "Golf". Buttons: "detail" and "My favourite is this!".
- ALTERNATIVE 5**: Switzerland Culture, History. Description: "Art Basel". Buttons: "detail" and "My favourite is this!".
- ALTERNATIVE 6**: Switzerland Alpine skiing, Snowboarding. Description: "Winter in Switzerland". Buttons: "detail" and "My favourite is this!".

Figure 4. Seeking Inspiration

3 TRIP@DEVICE RECOMMENDATION TECHNOLOGY

Trip@device is a recommendation technology which integrates case base reasoning and cooperative query answering to provide personalized suggestions about interesting tourist products. This section briefly describes the case structure and the methodology exploited in the two implemented recommendation functions, Single Item Recommendations and Seeking for Inspiration. Full details of Trip@device recommendation methodology can be found in [11].

3.1 Case Model

Trip@device bases its recommendations on a case model that captures a unique human-machine interaction session. A case collects: information provided by the user during the session, the products selected, and some long-term preferences and demographic data of the user if he is registered. Recommendation sessions are stored as cases in a repository (case base) [6]. In addition to the case base, product catalogues are stored in a relational database.

In Trip@device, a case represents a user interaction with the system, and it is built incrementally during the recommendation session. A case comprises the following three main components:

- **Collaborative Features (clf)** are features that describe general user's characteristics, wishes, constraints or goals (e.g. desire to relax or to practise sports). They capture general travel preferences and characteristics that are relevant to the decision-making process, but cannot be considered as product features (e.g. the traveller group composition). These features are used to measure case similarity.
- **Content Queries (cnq)** are queries posed over the catalogues of products. Content queries are built by constraining (content) features, i.e., descriptors of the products listed in the catalogues.
- **Cart** contains the set of products and services chosen by the user during the recommendation session. A cart represents a meaningful (from the user's point of view) bundling of different products. For instance, a travel cart may contain some destinations, some accommodations, and some additional attractions.

Figure 5 shows a case example. It represents a user, who is single, has a medium budget, interested in nature and culture and in visiting Scandinavia or Central Europe. These are the collaborative features. Then there is one query, where the user has further specified some countries he is interested in (Norway and Sweden). From the result set of that query, the user has selected the Raindeer sledding and the Trono old church site.

3.2 Single Item Iterative Selection

The overall process supported by Trip@device for the single item iterative selection is shown in Figure 6. The full description of this process can be found in [11], here we provide just a brief summary of the methodology. The user interacts with the recommender system by asking for recommendations about a product type (e.g., an activity)(1: AskRecommendation(q) in Figure 6). The system replies to this query q either recommending some products, or, in case the query fails, suggesting some query refinements. The RecEngine module manages the request. First, it invokes the Evaluate-Query function (2) of the Intelligent Query Manager module (IQM), by passing the query. This function searches the catalogue for products matching the query. If too many or no product matches the input

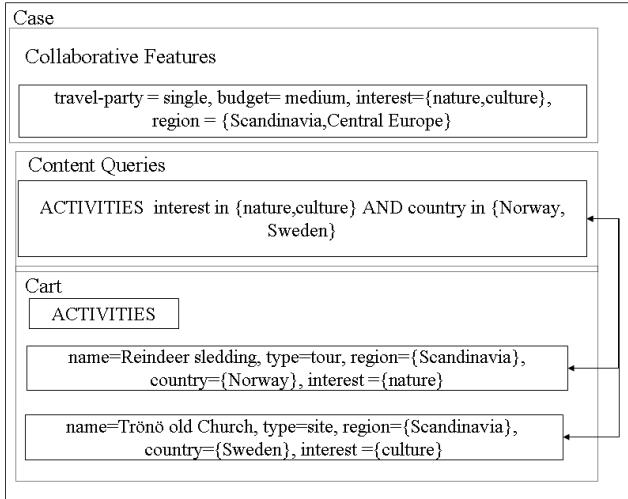


Figure 5. Example of a case

query q , then IQM analyzes q and determines a set of query refinements to suggest.

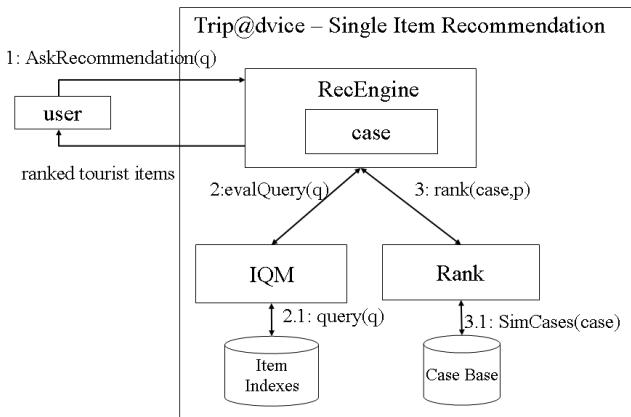


Figure 6. Single item recommendation with Trip@device.

The products selected by the user's query are ranked by invoking the Rank method (3). Rank receives as input the current case and the set of products p to be ranked. The current case is used to retrieve the set of K most similar cases from the case base, and the products contained in these retrieved cases are then used to rank the user selected products p . Finally, the ranked products are returned to the user. Ranking is computed exploiting two similarity metrics: first, the case base is accessed to retrieve the K most similar cases (reference cases) to the current one. Then, the products contained in the carts of the retrieved reference cases (reference products) are used to sort the products selected by the user's query. The basic idea is that among the products in the result set of the query one will get a better score if it is similar/equal to a product chosen by a user with similar needs and preferences. This function implements a hybrid "cascade" recommendation methodology [3], where first a conversational approach is used to select a set of options and then a collaborative via content approach is used for ranking [8].

3.3 Seeking for Inspiration

In this section we describe the seeking for inspiration function, designed for users that would prefer to browse travel options and get inspired by them before taking some decision. The recommendation proceeds according to the following loop (see Figure 7):

- **Retrieval.** The process is started with an initial case c that could be either the current case or a random case taken from the case base. It is the current user's case if the user has already created it in previous stages of the interaction, for instance by specifying his general travel preferences or adding some tourist products to the cart. Conversely, if the current case does not yet contain any useful information, then the recommender does not know anything about the user and could only make an initial random guess of what could be a good travel for the user. The retrieval module searches for the M most similar cases in the case base and passes this set to the next module, the Selection.
- **Case Selection.** In the second step the M cases retrieved from the case base are analyzed to select a very small subset of candidates to be presented to the user (six in the European tourism destination portal). To select this small set of candidates a greedy algorithm is used [14]. It iteratively selects a case starting from the initial case and puts the selected case in the result set. The case added at each iteration is the one that minimizes the sum of the similarities between itself and the cases already in the result set. In this way, it selects from the M cases those that are enough different among themselves to provide a reasonable variety of options to the user.

The selected cases are shown to the user, who can choose the preferred one. The retrieval step is executed again, with some notable changes. The seed case is now the case that received positive feedback from the user, and the number of cases retrieved from the case base, M in the first retrieval, is decreased by a factor $0 < \lambda < 1$. The rationale for decreasing the number of retrieved cases is to stronger focus the retrieval around the input case, since at this new stage we must count on the positive evaluation of the user on the selected case. This recommendation functionality follow the recommendation by proposing and comparison-based recommendation approaches introduced in [13, 7]

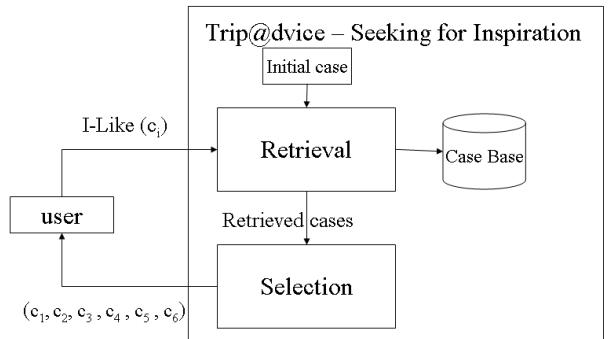


Figure 7. Seeking for inspiration recommendation cycle.

4 INTEGRATION ARCHITECTURE

As already said in the Introduction, integrating the recommendation technology in a full portal presents several issues that should be considered. [9] describes several options for deploying a recommender

system. One, is to build an independent web site, which provides recommendations about products described in a second portal. TripAdvisor is a popular example of this approach, it recommends hotels that are sold by a number of merchant web sites (e.g. expedia and travelocity). Another approach consists of deploying the recommendation technology inside the system itself, and tightly integrate the recommendation functions with the full system. In this way the designer can mix new recommendation functions with more traditional one (e.g. browsing and search), controlling the full human-computer interaction. In visiteurope.com we adopted the second option. Figure 8 shows a high level view of the modules relevant to our discussion. In particular it shows some components which are parts of a typical web portal architecture: the Navigation component, allowing the user to navigate the content of the portal following a navigation tree; the Content Management System (CMS), which is in charge of managing the content items of the portal; and TripAdvisor, providing the recommendation and travel plan management functions and the Travel Planner graphical user interface.

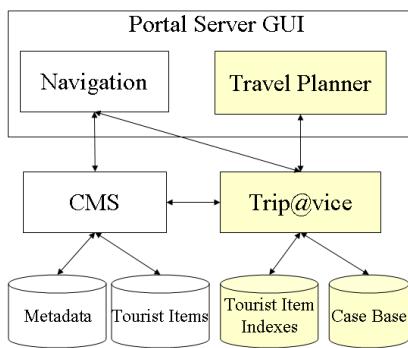


Figure 8. Single item recommendation with TripAdvisor.

These components should communicate among them. The Travel Planner GUI uses the TripAdvisor functions to display the current user travel plan and recommend items to the users. The Navigation service interacts with TripAdvisor to add to the user' travel plan the tourist items that the user chooses navigating in the portal or exploiting the recommendation functions. In this architecture, TripAdvisor must retrieve from the CMS the content items and their metadata to build the recommendations and must refer to them when storing cases.

5 CONCLUSION

In this paper we have illustrated the design and the implementation of a decision aid tool aimed at supporting the traveller's decisional process in the European tourism destination portal www.visiteurope.com. We have presented the main project goals, the interaction we have supported, its methodology and the integration architecture. The proposed recommendation technology, before being deployed in visiteurope.com, has been evaluated in a number of empirical studies involving hundreds of users [16]. The travel plans case base has been initialized with some cases for each country and is now growing, acquiring new travel cases as the system is used by the visitors. A case management tool, supporting the content manager in the validation and selective inclusion of new cases in the repository, has been developed and integrated in the portal infrastructure.

A number of open problems must be still faced in the near future. Techniques to analyze the case base and select a subset of cases among the potentially huge set of those created by the portal visitors must be developed. Moreover, tourism organizations should be supported in better understanding their users and their tourism offers by exploiting the knowledge stored in the case base.

REFERENCES

- [1] Gediminas Adomavicius and Alexander Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering*, **17**(6), 734–749, (2005).
- [2] Dimitrios Buhalis, *eTourism: Information Technology for Strategic Tourism Management*, Prentice Hall, 2003.
- [3] Robin Burke, 'Hybrid recommender systems: Survey and experiments', *User Modeling and User-Adapted Interaction*, **12**(4), 331–370, (2002).
- [4] K. Gräbler and A.H. Zins, 'Vacation trip decision styles as basis for an automated recommendation system: Lessons from observational studies', in *Proceedings of the ENTER 2002 Conference*, Innsbruck, Austria, (January 22-25 2002). Springer Verlag.
- [5] Y.-H. Hwang, U. Gretzel, and D. R. Fesenmaier, 'Behavioral foundations for human-centric travel decision-aid systems', in *Proceedings of the ENTER 2002 Conference*, Innsbruck, Austria, (January 22-25 2002). Springer Verlag.
- [6] Fabiana Lorenzi and Francesco Ricci, 'Case-based recommender systems: A unifying view', in *Intelligent Techniques for Web Personalization, IJCAI 2003 Workshop, ITWP 2003, Acapulco, Mexico, August 11, 2003, Revised Selected Papers*, pp. 89–113. Springer, (2005).
- [7] Lorraine McGinty and Barry Smyth, 'Comparison-based recommendation', in *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, eds., S. Craw and A. Preece, pp. 575–589, Aberdeen, Scotland, (4 - 7 September 2002). Springer Verlag.
- [8] Michael J. Pazzani, 'A framework for collaborative, content-based and demographic filtering', *Artificial Intelligence Review*, **13**, 393–408, (1999).
- [9] Ulrich Rabanser and Francesco Ricci, 'Recommender systems: Do they have a viable business model in e-tourism?', in *Information and Communication Technologies in Tourism 2005*, Springer Verlag, Wien - New York, (2005).
- [10] Francesco Ricci, 'Travel recommender systems', *IEEE Intelligent Systems*, **17**(6), 55–57, (2002).
- [11] Francesco Ricci, Dario Cavada, Nader Mirzadeh, and Adriano Venturini, 'Case-based travel recommendations', in *Destination Recommendation Systems: Behavioural Foundations and Applications*, eds., D. R. Fesenmaier, H. Werthner, and K. W. Wober, CABI Publishing, (2006).
- [12] J. Ben Schafer, Joseph A. Konstan, and John Riedl, 'E-commerce recommendation applications', *Data Mining and Knowledge Discovery*, **5**(1/2), 115–153, (2001).
- [13] Hideo Shimazu, 'ExpertClerk: Navigating shoppers buying process with the combination of asking and proposing', in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001*, ed., Bernhard Nebel, pp. 1443–1448, Seattle, Washington, USA, (August 4-10 2001). Morgan Kaufmann.
- [14] B. Smyth and P. McClave, 'Similarity vs diversity', in *Proceedings of the 4th International Conference on Case-Based Reasoning*, Springer-Verlag, (2001).
- [15] Hannes Werthner and Stefan Klein, *Information Technology and Tourism - A Challenging Relationship*, Springer, 1999.
- [16] Andreas H. Zins and Ulrike Bauernfeind, 'Evaluating travel recommender systems: A case study of dietorecs', in *Destination Recommendation Systems: Behavioural Foundations and Applications*, eds., D. R. Fesenmaier, H. Werthner, and K. W. Wober, CABI Publishing, (2006).

Natural and Intuitive Multimodal Dialogue for In-Car Applications: The SAMMIE System

Tilman Becker¹ and Nate Blaylock² and Ciprian Gerstenberger² and Ivana Kruijff-Korbayová² and Andreas Korthauer³ and Manfred Pinkal² and Michael Pitz⁴ and Peter Poller¹ and Jan Schehl¹

Abstract.

We present SAMMIE, a laboratory demonstrator of an in-car showcase of a multimodal dialogue system developed in the TALK project⁵ in cooperation between DFKI/USAAR/BOSCH/BMW, to show natural, intuitive mixed-initiative interaction, with particular emphasis on multimodal turn-planning and natural language generation. SAMMIE currently supports speech-centered multimodal access for the driver to a MP3-player application including search and browsing, as well as composition and modification of playlists. Our approach to dialogue modeling is based on collaborative problem solving integrated with an extended Information State Update paradigm. A formal usability evaluation of a first baseline system of SAMMIE by naive users in a simulated environment yielded positive results, and the improved final version will be integrated in a BMW research car.

1 Introduction

The TALK project investigates issues in multimodal dialogue systems: multilinguality, adaptivity and learning, dialogue modeling and multimodal turn planning. Our approach is based on an extended Information State Update paradigm. Some of these issues are demonstrated in SAMMIE, an in-car showcase developed in cooperation between DFKI/USAAR/BOSCH/BMW. The design of the SAMMIE system is based on a series of user studies performed in different Wizard-of-Oz settings as well as a usability evaluation of a baseline version of the laboratory demonstrator. SAMMIE will be integrated into a test car at BMW later this year.

The SAMMIE system provides a multimodal interface to an in-car MP3 player through speech and haptic input with a BMW iDrive input device, a button which can be turned, pushed down and sideways in four directions. System output is provided by speech and a graphical display integrated into the car's dashboard. An example of the system display is shown on the right in figure 1.

The MP3 player application offers a wide range of tasks: The user can control the currently playing song, search and browse the database by looking for any of the fields in the MP3 database (song, artist, album, etc.), search and select playlists and even construct and

edit playlists.

SAMMIE supports natural, intuitive mixed-initiative interaction, with particular emphasis on multimodal turn-planning and natural language generation. The system puts the user in control of the interaction. Input can be given through any modality and is not restricted to answers to system queries. On the contrary, the user can provide new tasks as well as any information relevant to the current task at any time. This is achieved through modeling the interaction as a collaborative problem solving process, modeling the tasks and their progression as *recipes* and a multimodal interpretation that fits any user input into the context of the current task. Note that the user is also free in the use of multimodal input, such as deictic references accompanied by pointing gestures ("Play this title" while pushing the BMW iDrive button), and even cross-modal references without pointing as in "Play the third song (on the list)". To support these aspects of dialogue flexibility, we model dialogue context, collaborative problem solving and the driver's attention state by an enriched information state. Table 1 shows a typical interaction with the SAMMIE system, figure 3 shows the current setup for the user environment.

- U: Show me the Beatles albums.
- S: I have these four Beatles albums. [shows a list of album names]
- U: Which songs are on this one? [selects the Red Album]
- S: The Red Album contains these songs [shows a list of the songs]
- U: Play the third one.
- S: [song "From Me To You" plays]

Table 1. A typical interaction with SAMMIE.

The following section describes our system architecture. Section 3 presents our approach to extended multimodal interaction modeling, ontology based modeling and its impact on natural and intuitive dialogues. Section 4 briefly describes our Wizard-of-Oz experiments, the evaluation process and the results. Finally, section 5 summarizes some important lessons learned in development and evaluation of our system.

2 System Architecture

Our system architecture follows the classical approach [5] of a pipelined architecture with multimodal fusion and fission modules encapsulating the dialogue manager. Figure 1 shows the modules and their interaction: Modality-specific recognizers and analysers provide semantically interpreted input to the multimodal fusion module (interpretation manager in fig. 1) that interprets them in the context of the other modalities and the current dialog context. The dialogue manager decides on the next system move, based on its model of the

¹ DFKI, Saarbrücken, Germany, email: First.Last@dfki.de

² Saarland University (USAAR), Saarbrücken, Germany, email: {blaylock,gerstenb,korbay,pinkal}@coli.uni-sb.de

³ Robert Bosch GmbH, Stuttgart, Germany, email: Andreas.Korthauer@de.bosch.com

⁴ BMW Group Forschung und Technik, München, Germany, email: Michael.Pitz@bmw.de

⁵ TALK (Talk and Look: Tools for Ambient Linguistic Knowledge) www.talk-project.org is funded by the EU as project No. IST-507802 within the 6th Framework program.

tasks as collaborative problem solving, on the current context and also on the results from calls to the MP3 database. The turn planning module then generates an appropriate message to the user by planning the actual content, distributing it over the available output modalities and finally co-ordinating and synchronizing the output. Modality-specific output modules generate spoken output and an update of the graphical display. All modules interact with the extended information state in which all context information is stored.

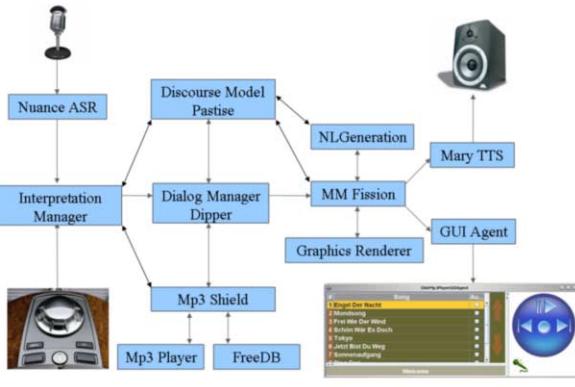


Figure 1. SAMMIE system architecture.

Many tasks in the SAMMIE system are modeled by a plan-based approach. Discourse modeling, interpretation management, dialogue management and linguistic planning, and turn planning in our system are all based on the production rule system PATE⁶ [9], originally developed for the integration of multimodal input. During the implementation of the baseline system, we found that PATE is also adequate for modeling other dialogue system components as it provides an efficient and elegant way of realizing complex processing rules. Section 3.5 elaborates more on PATE and the ontology-based representations it uses. In the following sections we will concentrate on our main areas of research.

3 Modeling Multimodal Interaction

Many dialogue systems that are employed today follow a state-based approach that explicitly models the full (finite) set of dialogue states and all possible transitions between them. The VoiceXML⁷ standard is a prominent example of this approach. This has two drawbacks: on the one hand, this approach is not very flexible and typically allows only so-called system controlled dialogues where the user is restricted to choosing their input from provided menu-like lists and answering specific questions. The user never is in control of the dialogue.⁸ For restricted tasks with a clear structure, such an approach is often sufficient and has been applied successfully. On the other hand, building such applications requires a fully specified model of all possible states and transitions, making larger applications expensive to build and difficult to test.

⁶ PATE is short for (P)roduction rule system based on (A)ctivation and (T)yped feature structure (E)lements.

⁷ <http://www.w3.org/TR/voicexml20>

⁸ It is possible to build state-based systems without presenting a fixed menu hierarchy to the user. The user might even get the impression that they can interact quite flexible. But of course this creates considerable application complexity and implementation effort for all possible states and transitions.

In SAMMIE, we are following an approach that models the interaction on an abstract level as collaborative problem solving and adds application specific knowledge on the possible *tasks*, available *resources* and known *recipes* for achieving the goals. A planner, based on the PATE rule-interpreter [9] dynamically derives the next move of the system and then plans the details of the system output.

In addition, all relevant context information is administered in a central Extended Information State (EIS) module. This is an extension of the Information State Update approach [13] to the multimodal setting.

3.1 Extended Information State

The information state of a multimodal system needs to contain a representation of contextual information about discourse, but also a representation of modality-specific information and user-specific information which can be used to plan system output suited to a given context. The overall information state (IS) of the SAMMIE system is shown in Figure 2.

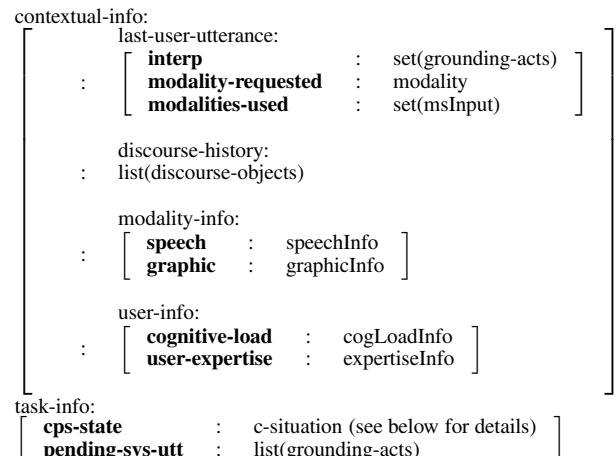


Figure 2. Structure of the SAMMIE Information State

The contextual information partition of the IS represents the multimodal discourse context, i.e., the information communicated through the different modalities. It contains a record of the latest user utterance and preceding discourse history representing in a uniform way the salient discourse entities introduced in the different modalities (we follow the approach adopted in the SmartKom system [10]; the discourse model employs the three-tiered context representation proposed in [7] where the linguistic layer is generalized to a modality layer). The contents of the task partition are explained in the next section.

3.2 Collaborative Problem Solving

We see a dialogue system as a *conversational agent*—an autonomous agent which can communicate with humans through natural language dialogue. In order to support natural and flexible conversation, we need to model dialogue about the range of activities an agent may engage in, including goal selection, planning, execution, monitoring, replanning, and so forth. To achieve this our dialogue manager is based on an agent-based model which views dialogue as collaborative problem-solving (CPS) [4]. CPS follows a process similar to

single-agent problem solving, but the agents negotiate to jointly determine objectives, find and instantiate recipes to accomplish them and execute the recipes and monitoring for success.

The basic building blocks of the formal CPS model are problem-solving (PS) objects, which we represent as typed feature structures. PS object types form a single-inheritance hierarchy, where children inherit or specialize features from parents. Instances of these types are then used in problem solving.

In our CPS model, we define types for the upper level of an ontology of PS objects, which we term *abstract PS objects*. There are six abstract PS objects in our model from which all other domain-specific PS objects inherit: objective, recipe, constraint, evaluation, situation, and resource. These abstract PS objects are used to model problem-solving at a domain-independent level and are taken as arguments by all update operators of the dialogue manager which implement conversation acts [4, 12]. The model is then specialized to a domain by inheriting and instantiating domain-specific types and instances from the PS objects. For example, the domain-specific objective *play-song* in the MP3 domain inherits all attributes from *objective* and adds a has-song slot, which is *single-slot* of type *song*. The operators, however, do not change with domain, which supports reasoning done at a domain-independent level.

3.3 Adaptive Turn Planning

In a multimodal dialogue system, the *fission* component is responsible for realising the planned system response as determined by the dialogue manager through multimodal output in an appropriate combination of the available output channels. This task comprises detailed content planning, media allocation and coordination and synchronization.

In SAMMIE, the fission task is realised by two modules - the *Turn Planner* and the *Output Manager*, whereas user- and modality-specific information which might be necessary for presentation planning can be obtained from another module called *Pastis*. Pastis is designed to provide and store user-, modality- and also discourse-specific information and forms, together with the dialogue manager's embedded information state as the *Extended Information State* of the system.

When the dialogue manager has finished processing the user input, the turn planner (TP) receives a bundle of CPS-specific conversational acts, representing the planned system response on an abstract level. TP then starts planning how to distribute given information over the available modalities, namely speech and graphics, but also determines on which level of detail information is going to be presented. As soon as TP has finished processing, it sends a sorted bundle of output messages, including both speech and graphic messages, to the output manager. The output manager then distributes the messages to the graphics renderer and/or the generation manager and synchronizes their output rendering.

The Turn Planner within the SAMMIE system is responsible for content selection and media allocation. It takes a set of CPS-specific conversational acts generated by the dialogue manager and maps them to modality-specific communicative acts. Therefore, relevant information on how content should be distributed over the available modalities (speech or graphics) can be obtained by

- (a) the EIS/discourse module Pastis, which provides information about
 - (1) the modality on which the user is currently focused, derived by the current discourse context.

- (2) the user's current cognitive load when system interaction becomes a secondary task (e.g., system interaction while driving). This information is modeled as a state variable with the possible states *low, mid, high* or *extreme*.
- (3) the user's expertise, which is represented as a state variable with the possible states *power-user* or *beginner*.

Pastis also contains information about factors that influence the preparation of output rendering for a modality, like the currently used language (German or English) or the display capabilities (e.g., maximum number of displayable objects within a table).⁹

- (b) a set of production rules that determine which kind of information should be presented through which modality. The rule set is divided in two subsets, domain-specific and domain-independent rules which together form the system's multimodal plan library. These rules are used to create the presentation plan for a turn, while dynamically taking the information listed in (a) into account.

Beside determining the best possible distribution of information, the turn planner also transforms domain-specific information that will be presented by speech to a representation we call *Reduced Knowledge Representation* (RKR) that can be interpreted by the Linguistic Planner. Such an RKR structure differs from its source structure through its more general, abstract surface structure which can be seen as an intermediate form between a pure ontological representation of a domain specific object and the logical form representation that the linguistic planner needs for deep generation with OpenCCG. Furthermore, an RKR structure specifies exactly the information that has to be presented to the user. Beside deriving the appropriate input for NLG, the turn planner is also responsible for computing and packaging the appropriate information to be presented by the display. Additionally, if there are alternatives on how to graphically realize content in different ways the turn planner needs to decide which one to take.

3.4 Spoken Natural Language Output Generation

Our goal is to produce output that varies in the surface realization form and is adapted to the context. We opted for using both a template-based and a deep grammar-based module in parallel. On the one hand, deep generation provides linguistically more powerful modeling, in particular it allows for more fine-grained and controlled choices between linguistic expressions in order to achieve contextually appropriate output. On the other hand, the template-based module can be developed faster and thus facilitates incremental development. It is also sufficient for classes of system output that do not need fine-tuned context-driven variation, such as simple cases of feedback. Our template-based generator can also deliver alternative realizations, e.g., alternative syntactic constructions *There are 3 songs by Nena* vs. *I found 3 songs by Nena*, referring expressions *Nena* vs. *the artist Nena*, or lexical items, e.g., *song* vs. *track*; however, the choice among alternative templates is made at random. The template-based generator is implemented by a set of straightforward sentence planning rules in the PATE system to build the templates, and a set of XSLT transformations to yield the output strings. Output in German and English is produced by accessing different dictionaries in a uniform way.

The grammar-based generator uses OpenCCG, an open-source natural language processing environment[3]. We have developed a

⁹ Note, that for points (2) and (3) we haven't yet fully elaborated the rule based processing of these factors as this part of our ongoing work.

German OpenCCG grammar with basic coverage of German phenomena, and gradually extend it with respect to the phenomena encountered in the SAMMIE-1 and SAMMIE-2 corpora (see section 4).

3.5 Modeling with an Ontology

We use a full model of the application in OWL¹⁰ format as the knowledge representation format in the dialogue manager, turn planner and sentence planner. This model includes the entities, properties and relations of the MP3 domain—including the player, data base and playlists. Also, all possible tasks that the user may perform are modeled explicitly. Note that this is a model that is *user centered* and not simply a model of the application’s API. Actually, there is a separate module, the MP3-shield, that maps user tasks into possibly complex interactions with the connected applications, i.e., the database and the MP3 player itself.

The OWL-based model is transformed automatically to the internal format used in the PATE rule-interpreter. PATE employs Typed Feature Structures (TFSs) as basic internal data representation and XML for encoding all incoming and outgoing data as well as knowledge bases (production rules, type definitions). PATE is based on some concepts of the ACT-R 4.0 system [2]. Its main concepts, which PATE makes use of, are the goal-oriented application of production rules, the activation of working memory elements, and the weighting of production rules. In processing TFSs, PATE provides two operations that both integrate data and also are suitable for condition matching in production rule systems, namely a slightly extended version of the general *unification*, but also the discourse-oriented operation *overlay* [1].

Another important feature is the concept of multiple inheritance provided by the type system, as it allows to define different views on ontological concepts. Consider the concept *Song* and the different views our system ontology provides. A *Song* can be seen as a *Browsable-object* which allows generalization within the turn planning library over objects a user can browse. It can also be seen as a *Media-object* or a *Problem-solving-object* which are abstract concepts dialogue management can use for planning and execution. Or it can be seen as a *Mp3-resource* which denotes the domain affiliation of the concept. Thereby PATE provides an efficient and elegant way to create more abstract/generic presentation planning rules.

4 Experiments and Evaluation

To guide system development we have so far conducted two WOZ *data collection* experiments and one *evaluation* experiment with a baseline version of our system. The SAMMIE-1 WOZ experiment involved only spoken interaction, SAMMIE-2 was multimodal, with speech and haptic input, and the subjects had to perform a primary driving task using a Lane Change simulator [8] in a half of the experiment session. The wizard was simulating an MP3 player application with access to a large database of information (but not actual music) of more than 150,000 music albums (almost 1 million songs). The user had to carry out several tasks of two types: searching for a title either in the database or in an existing playlist, and building a playlist satisfying a number of constraints. In order to collect data with a variety of interaction strategies, we used multiple wizards and gave them freedom to decide about their response and its realization. In the multimodal setup in SAMMIE-2, the wizards could also freely decide between mono-modal and multimodal output. There is not

enough time for the wizard to properly design the screen output on the fly. Therefore, we implemented modules supporting the wizard by providing several automatically generated screen output options the wizard could select from and inform the user about the database search results using the visual modality.

The following aspects of the setup were designed to elicit interactions more realistically resembling dialogue with an actual system [6]: The wizard and the user did not directly hear each other, instead, their utterances were immediately transcribed; the wizard’s utterances were then presented to the user via a speech synthesizer, and parts of the user’s utterances were sometimes deleted to simulate acoustic understanding problems and elicit clarifications.



Figure 3. The current setup of the user environment.

To explore the user acceptance, usability, and performance of a first baseline implementation of the SAMMIE multimodal dialogue system we have completed a usability evaluation. The evaluation tested the multimodal interaction of first-time users with the SAMMIE system in a laboratory experiment with a simulated driving task (figure 3 shows the setup). A sample of 20 subjects performed 32 tasks in total out of three scenarios with variation of the different kinds of dialogue modalities (spoken vs. multimodal). The users were asked to perform tasks which tested the system functionality, namely controlling player functions like stop/play/pause, next/previous track, playing a particular/random song/album/playlist, querying the music database (e.g., available songs/albums/artists/playlists) and administration of playlists (create/delete playlists, add/remove songs).

The evaluation analyzed the user’s interaction with the baseline system and combined objective measurements like task completion and speech recognition accuracy observed in the experiments and subjective ratings from the test subjects by means of intermediate interviews during the session and by post-experimental questionnaires. The analysis of the experiments yielded an overall task completion rate (TCR) of about 90% for both spoken and multimodal interaction. Note that we allow up to four repeats for a user input. Relatively small differences in TCR between the dialogue modalities but considerable decrease of TCR for the more complex tasks have been observed. This was partly due to a relatively high out-of-grammar rate and consequently word error rate which showed the need to further increase the ASR grammar coverage.

Spoken dialogue was observed as the preferred interaction modality in the experiments. About 70% of the subjects chose speech when they had the free choice and less than 10% changed the modality during the task. Nevertheless, 40% of all subjects would prefer multimodal interaction in the long run when having more practice with the system. The general impression of the SAMMIE system was rated positively by most of the test subjects and changing the interaction modality was simple or very simple for 95% of the users. Also, the content and extent of the system’s spoken and graphical output messages were rated mostly positively. But a detailed analysis of the ob-

¹⁰ <http://www.w3.org/TR/owl-features>

jective measurements and subjective ratings also revealed some important issues for significant improvement towards the final version of the SAMMIE system.

The following section reflects some of the lessons learned from this evaluation.

5 Lessons Learned

To summarize the current state of our research and development, this section attempts to present the most important lessons learned to date and the biggest challenges for AI in the domain of natural, intuitive multimodal interaction.

- Reliable and robust ASR: When the system does not recognize the user's utterance, the biggest challenge for the user and the system is to know what went wrong. We are addressing this on two fronts: SAMMIE tries to give feedback about partially understood utterances with appropriate clarification requests. Also, we are using the data collection to expand the language covered by the recognition grammars.
- Natural and understandable speech synthesis: We have used the multilingual capabilities of the Mary TTS [11] to pronounce English song titles correctly, even when embedded in a German utterance. Also, markup of the speech output should allow context dependent prosodic cues such as pauses around elements to be clarified. To address users' demands, we need to increase the naturalness and acoustic quality of the TTS.
- System responsiveness: For command and control type interaction, e.g., stopping the MP3 player, time delays should be in the millisecond range. Thus we have added shortcuts directly from speech recognition to the player application, such that the full dialogue system is updated only in parallel or later. However, the context model must be kept synchronized with the course of the dialogue. Overall, reaction times (for complex input) have to be improved to satisfy the requirements of in-car systems.
- Close to real time system feedback: early on, we have added a multimodal microphone state, signaling to the user whether the system is ready and listening or currently processing input. The microphone shown on the GUI (see figure 1) toggles between red and green, accompanied by characteristic acoustic signals.
- Speech-centered multimodality: The evaluation confirmed that speech is the most important modality for our system, however the graphical and haptic modalities were used and valued by users. Since driving as the primary task leaves fewer and sometimes little attention (in particular, visual attention) for interacting with the SAMMIE system, turn planning must assure that core information is conveyed in speech and only additional information is presented on the display. Such additional information must also be accessible through spoken interaction.
- Adaptive, context-sensitive presentation: Natural, intuitive, multimodal interaction can be achieved through true mixed-initiative, collaborative interaction. Our system dynamically adapts its next move to give "intelligent" replies that (i) make the system's understanding transparent and (ii) ask for clarification when necessary and (iii) query for more information by dynamically determining the most informative type of information, e.g. album name rather than artist name¹¹.

The last point represents the ultimate goal of our work: optimize all the above issues in the context of a flexible dialogue paradigm

(Extended Information State Update) to achieve natural and intuitive multimodal interaction.

6 Conclusion

We presented an in-car multimodal dialogue system for an MP3 application developed in the TALK project in cooperation between several academic and industrial partners. The system employs the Information State Update paradigm, extended to model collaborative problem solving and multimodal context. It supports natural, intuitive mixed-initiative interaction, with particular emphasis on multimodal turn-planning and natural language generation to produce output adapted to the context, including the driver's attention state with respect to the primary driving task. We performed extensive user studies in a WOZ setup to guide the system design. A formal usability evaluation of the system's baseline version in a simulated environment has been carried out with overall positive results. A further enhanced version of the system will be integrated and evaluated in a test car, demonstrating the successful transfer of a range of AI techniques towards a real world application.

REFERENCES

- [1] J. Alexandersson and T. Becker, 'Overlay as the basic operation for discourse processing in a multimodal dialogue system', in *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Seattle, Washington, (August 2001).
- [2] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*, Lea, June 1998.
- [3] J. M. Baldridge and G.J. M. Kruijff, 'Multi-Modal Combinatory Categorial Grammar', in *Proc. of the 10th Annual Meeting of the European Chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary, (April 2003).
- [4] N. Blaylock and J. Allen, 'A collaborative problem-solving model of dialogue', in *Proc. of the 6th SIGdial Workshop on Discourse and Dialogue*, eds., L. Dybkjær and W. Minker, pp. 200–211, Lisbon, (September 2–3 2005).
- [5] H. Bunt, M. Kipp, M. Maybury, and W. Wahlster, 'Fusion and coordination for multimodal interactive information presentation: Roadmap, architecture, tools, semantics', in *Multimodal Intelligent Information Presentation*, eds., O. Stock and M. Zancanaro, volume 27 of *Text, Speech and Language Technology*, 325–340, Kluwer Academic, (2005).
- [6] I. Kruijff-Korabayová, T. Becker, N. Blaylock, C. Gerstenberger, M. Käfser, P. Poller, J. Schehl, and V. Rieser, 'An experiment setup for collecting data for adaptive output planning in a multimodal dialogue system', in *Proc. of ENLG*, (2005).
- [7] S. LuperFoy, *Discourse Pegs: A Computational Analyses of Context Dependent Referring Expressions*, Ph.D. dissertation, University of Texas at Austin, December 1991.
- [8] S. Mattes, 'The Lane-Change-Task as a tool for driver distraction evaluation', in *Proceedings of IGfA*, (2003).
- [9] N. Pfleger, 'Context based multimodal fusion', in *ICMI '04: Proc. of the 6th international conference on Multimodal interfaces*, pp. 265–272, New York, NY, USA, (2004). ACM Press.
- [10] N. Pfleger, J. Alexandersson, and T. Becker, 'A robust and generic discourse model for multimodal dialogue', in *Proceedings of the 3rd Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, (2003).
- [11] M. Schröder and J. Trouvain, 'The german text-to-speech synthesis system MARY: A tool for research, development and teaching', in *The Proceedings of the 4th ISCA Workshop on Speech Synthesis, Blair Atholl, Scotland*, (2001).
- [12] D. R. Traum and E. A. Hinkelman, 'Conversation acts in task-oriented spoken dialogue', *Computational Intelligence*, 8(3), 575–599, (1992). Also available as University of Rochester Department of Computer Science Technical Report 425.
- [13] D. R. Traum and S. Larsson, 'The information state approach to dialog management', in *Current and New Directions in Discourse and Dialog*, Kluwer, (2003).

¹¹ We are currently experimenting with clustering algorithms to adequately summarize large sets of answers for a database query.

A Client/Server User-Based Collaborative Filtering Algorithm: Model and Implementation

CASTAGNOS Sylvain and BOYER Anne¹

Abstract. This paper describes a new way of implementing an intelligent web caching service, based on an analysis of usage. Since the cache size in software is limited, and the search for new information is time-consuming, it becomes interesting to automate the process of selecting the most relevant items for each user. We propose a new generic model based on a client/server collaborative filtering algorithm and a behavior modeling process. In order to highlight the benefits of our solution, we collaborated with a company called ASTRA which is specialized in satellite website broadcasting. ASTRA has finalized a system sponsored by advertisement and supplying to users a high bandwidth access to hundreds of websites for free. Our work has been implemented within its software architecture and, in particular, within its recommender system in order to improve the satisfaction of users. Our solution is particularly designed to address the issues of data sparsity, privacy and scalability. Because of the industrial context, we consider the situation where the set of users is relatively stable, whereas the set of items may vary considerably from an execution to another. In addition to the model and its implementation, we present a performance assessment of our technique in terms of computation time and prediction relevancy.

1 INTRODUCTION

With the development of information and communication technologies, the size of information systems all over the world has exponentially increased. This causes at least two problems. First, it becomes difficult for users to identify interesting items in a reasonable time, even if they use a powerful search engine. Secondly, the amount of available data is now much more important than the cache size of some industrial applications on client side. We can quote the examples of strategic awareness tools in big companies or the satellite data transfers. In this paper, we propose a generic collaborative filtering model to cope with these problems. Our approach is suitable for client/server architectures. We designed it to address three main difficulties inherent in an industrial context: the recommender system scalability, the user privacy and the data sparsity.

We implemented our work in the context of satellite website broadcasting. Thus, we run our model in the architecture of Casablanca, which is a product of the ASTRA company². ASTRA conceived a service of satellite website broadcasting service called Sat@once. This service is sponsored by advertisement so that it is free for users, provided that they use a DVB receiver. The satellite bouquet holds 150 websites which are sent to about 120.000 persons through a high-bandwidth and one-way transmission. More-

over, users can inform the server about their preferences by using a standard internet connection. Initially, they only could send the non-numerical votes which appear as a list of favorite websites. Once a week, the preferences of all users are used to make up the bouquet with the most popular websites. Because of this industrial context, we consider the situation where users are relatively stable, whereas the set of items may vary considerably from an execution to another. In addition to the research constraints, we also had to be compliant with the Casablanca architecture and the ASTRA business model.

The next section is dedicated to a short state-of-the-art about collaborative filtering techniques. Afterwards, we will present our model and its implementation in a satellite website broadcasting software. The fourth part offers an evaluation of our algorithm, both in terms of computation time and relevancy recommendations. At last, we will highlight the benefits of this decentralized architecture.

2 RELATED WORK

BREESE *et al* [3] have identified, among existing techniques, two major classes of algorithms to solve this problem : the memory-based and the model-based algorithms.

The memory-based algorithms maintain a database containing votes of all users. A similarity score (RENNICK [9], MAES [11] or BREESE [3]) is determined between active user and each of the others members. Then, each prediction leads to a computation on all of this source of data. The influence of a person is all the stronger in it since his/her degree of similarity with the active user is great.

These memory-based techniques offer the advantage to be very reactive, by integrating immediately modifications of users profiles into the system. However, BREESE *et al* [3] are unanimous in thinking that their scalability is problematic: even if these methods work well with small-sized example, it is difficult to change to situations characterized by a great number of documents or users. Indeed, time and space complexities of algorithms are much too important for big databases.

The model-based algorithms constitute an alternative to the problem of combinatorial complexity. In this approach, collaborative filtering can be seen as the computation of the expected value of a vote, according to preferences of active user. These algorithms create descriptive models correlating persons, resources and associated votes via a learning process. Then, predictions are inferred from these models. The item-item algorithm [10] belongs, among others, to this family. UNGAR and FOSTER [12] have also imagined the possibility to classify users and resources in groups. So, for each category of users, we must estimate the probability that a resource could be chosen.

According to PENNOCK *et al* [8], model-based algorithms minimize the problem of algorithmic time complexity. Furthermore, they

¹ LORIA - Université Nancy 2, B.P.239, 54506 Vandoeuvre-lès-Nancy Cedex, France, email: {sylvain.castagnos, anne.boyer}@loria.fr

² <http://www.ses-astra.com/>

perceive in these models an added value beyond the only function of prediction: they highlight some correlations in data, proposing by that way an intuitive reasoning for recommendations or making simply the hypotheses more explicit. However, these methods are not enough dynamic and they react badly to insertion of new contents in database. Moreover, they require a learning phase at the same time penalizing for the user³ and expensive in computation time for large databases.

Consequently, one of the main difficulty of collaborative filtering remains the scalability of systems. The model proposed in this article is an hybrid approach, combining advantages of memory-based and model-based methods to distribute the computations between the server and the clients.

3 ARCHITECTURE

The architecture of our information filtering system is shown on figure 1. This model associates a behavior modeling method based on the Chan formula [5] and a new version of the hierarchical clustering algorithm [6]. This new version – called FRAC in the rest of this article – has the advantages to be distributed, to be optimized in computation time and to converge systematically.

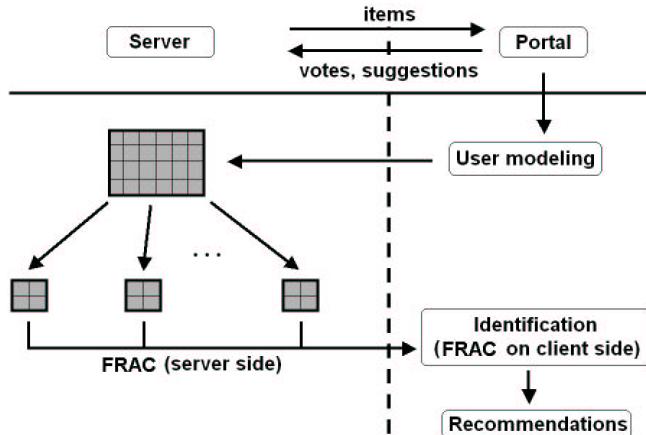


Figure 1. Architecture of the information filtering module.

In order to distribute the system, the server side part is separated from the client side. The function of user modeling determines numerical votes for items according to user actions. Then, numerical votes are sent to the server, like the non-numerical ones⁴. Thus, the server uses, as input parameters, the matrix of user votes and the database including sites and descriptors. In this way, the server has no information about the population, except anonymous votes. User preferences are stored in the profile on clients. Thus, the confidentiality criterion is duly fulfilled.

The FRAC algorithm aims at reducing quantity of data that must be processed. The offline computations of FRAC allow to build typical user profiles. It is no longer necessary to consider the whole vote matrix, but only votes of persons belonging to the group of the active user. This reduces the number of considered people, but also

³ The recommendations system won't be able to provide relevant documents as soon as it receives the first queries.

⁴ The list of favorites are given explicitly by users, while numerical votes are estimated in a transparent way. This is the reason why we use the non-numerical votes to determine the content of the bouquet.

the number of items: it is pointless to keep documents that none of the group members has read. In this way, we avoid the problem of bottleneck of collaborative filtering on client side: the active user is identified to one of the typical users groups in a very short time.

3.1 Behavior modeling

In the Casablanca interface, users have the possibility to define a list of favorites, that is to say some sites they are interested in among those contained in the bouquet. However, we can't describe these non-numerical votes as boolean. We can't differentiate items in which the active user is not interested (negative votes) from those he/she doesn't know or has omitted. This kind of votes is not sufficient to do relevant predictions with collaborative filtering methods.

In an ideal recommender system, users can explicitly rate each item. Ratings are integers in accordance with an arbitrary graduation. Nevertheless, in an industrial context, there can be marketing reasons which require to proceed differently. The ASTRA satellite bouquet is partially fulfilled with websites paid by advertisers. In this case, the risk that most users negatively and explicitly rate an advertisement can't be taken. For this reason, we have chosen to determine numerical marks in a transparent way for users.

Another advantage of this method is to deal with the problem of sparsity by increasing the number of votes in the matrix. Users generally rate the websites with whom they are familiar. It only represents a small part of the available items. The behavior modeling process is based on implicit criteria. A function estimates ratings that the active user is likely to give to different websites. It relies on several parameters, such as the percentage of visited links, the time or the reading frequency. In this way, the system can collect much more information than when using explicit ratings of users.

Casablanca generates log files on client side. These files are very similar to those of an Apache server. Each time a page is loaded in the portal, pieces of data are stored including time, path of the page and information about the website it belongs to. We consequently deduce the user preferences from an analysis of usage.

To do so, we developed a user modeling function based on the formula of Philip CHAN [5]. We adapted this formula in such a way that it is applicable to items (cf. infra, formula 1, p. 3), while the original formula was designed for web pages. But, in the ASTRA context, items correspond to websites, that is to say indivisible sets of pages. The time spent on an item is, for example, calculated as the cumulative times spent on each of its pages. A consultation is the set of consecutive lines relating to a same website in the log files. The log files only stored the uploading time of a page. We consequently have to limit the duration beyond the last upload in a consultation, if there is no other consultation just after. We also modified coefficients in the original Chan formula, in order to optimize the results in accordance with log files of ASTRA.

All pieces of information retrieved in the log files remain on client side, in order to preserve privacy. Only numerical votes which have been deduced from this process are sent anonymously to the server⁵. We call them "user profiles". They are required for the use of FRAC clustering algorithm. These profiles are not persistent, that is to say stored for a limited duration. Moreover, the sending on server side is initiated by the active user.

⁵ There is no way to retrieve the user from a profile on the server [4].

Adapted Chan formula:

$$\text{Interest}(\text{item}) = 1 + 2 \cdot \text{IsFavorite}(\text{item}) + \text{Recent}(\text{item}) + 2 \cdot \text{Frequency}(\text{item}) \cdot \text{Duration}(\text{item}) + \text{PercentVisitedLinks}(\text{item})$$

$$\text{With: } \text{Recent}(\text{item}) = \frac{\text{date(last visit)} - \text{date(log beginning)}}{\text{date(present)} - \text{date(log beginning)}}$$

$$\text{And: } \text{Duration}(\text{item}) = \max_{\text{consultations}} \left(\frac{\text{time spent on pages of item}}{\text{size of the item}} \right) \quad (1)$$

$\text{Interest}(\text{item})$ must be rounded up to the nearest integer.

$\text{IsFavorite}(\text{item})$ equals 1 if the item has been voted by the user (non-numerical vote) and 0 otherwise.

$\text{Frequency}(\text{item}) \cdot \text{Duration}(\text{item})$ must be normalized so that the maximum is 1.

$\text{PercentVisitedLinks}(\text{item})$ corresponds to the number of visited pages divided by the number of pages on the item.

3.2 Clustering algorithm

Once the profiles of users have been sent to the server, the system has to build virtual communities of interests. In our model, this step is carried out by a hierarchical clustering algorithm, called FRAC. This is an improved and decentralized version of the RecTree algorithm [6]. It attempts to split the set of users into cliques by recursively calling the nearest neighbors method [7] (K-Means), as illustrated on figure 2.

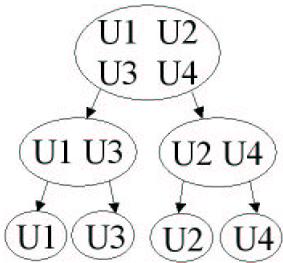


Figure 2. Hierarchical organization of users.

In this section, we explain how to build typical user profiles on server side and how to identify the active user to a group. This second step takes place on client side. We made sure that the identification phase is done in a very short response time. In this way, the client part provides real-time predictions. Offline computation time on server side has also been improved to fit in with the update frequency of the ASTRA satellite bouquet.

The FRAC algorithm is a model-based approach, described as a clustering method. However, it is managed as a memory-based approach because all the pieces of information are required for similarity computation. It allows, within the scope of our architecture, to limit the number of persons considered in the prediction computations. Thus, the results will be potentially more relevant, since observations will be based on a group closer to the active user [12].

In order to compute these groups of interests, the server extracts data from the profiles of users and aggregates the numerical votes in a global matrix. This matrix constitutes the root of the tree (cf. supra, fig. 2). The set of users is then divided into two sub-groups using the K-means method. In our case, the number k equals 2, since our overall strategy is to recursively divide the population into binary sub-sets. Once this first subdivision has been completed, it is repeatedly applied to the new subgroups, and this until the selected depth of the tree has been reached. This means, the more one goes down in the structure of the tree, the more the clusters become specific to a certain group of similar users. Consequently, people belonging to a

leaf of the tree share the same opinion concerning the assignment of a rating for a given item.

The K-Means algorithm is very sensitive to initial starting conditions and may converge more or less quickly to different local minima⁶. The usual way to proceed consists in choosing randomly k centers in the users/items representation space. Numerous studies have been made to improve K-Means by refining the selection of these initial points [2]. But it remains a difficult problem and some of these approaches don't obtain better results than the method with a random initialization. In our case, the problem is much simpler since we only have two centers. Thus, we propose a new way to select the starting points, shown on figure 3.

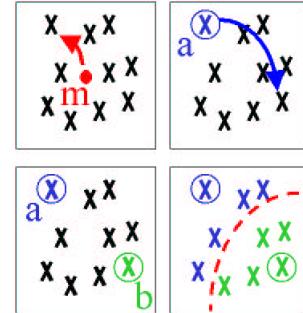


Figure 3. Initialization of 2-Means algorithm.

We work in a N-dimensional space, since the coordinates correspond to the votes of users for the N items. However, the example on the diagram 3 is in dimension 2 for more legibility. We start from the principle that the two most distant users are inevitably in different clusters. Consequently, they constitute the ideal candidates for the initial points. To identify them, we first search the most distant point from the middle M of the users/items representation space. This point is called A on figure 3. Then, we compute the point B, which is the most distant from A. A and B are subsequently the starting points of the 2-Means algorithm. This initialization phase is in $O(2n)$, where n is the number of users. Metrics used to determine distances is the Pearson correlation coefficient [9].

Once groups of persons have been formed as previously mentioned, the position of the center is recalculated for each cluster (either by computing the isobarycenter, either by using the equation 2 according to the precision we want) and this operation is repeated from the beginning until we have obtained a stable state (where the

⁶ We want to minimize the distances between users of a same group and maximize them between users of different clusters.

centers no longer move after recalculation of their position). Our initialization allows to reach this state much more quickly.

$$r_{c_{t+1},l} = \alpha \cdot \sum_{Y_{c_t,u}} (r_{u,l} \cdot |w(c_t, u)|) \quad (2)$$

With: $r_{c_{t+1},l}$ the coordinate of the new center c_{t+1} for the item l;
 $r_{u,l}$ the vote of the user u for the item l;
 $w(c_t, u)$ the distance between the previous center and u;
 $Y_{c_t,u} = \{u | w(c_t, u) \neq 0\}$;
 $\alpha = \frac{1}{\sum_{Y_{c_t,u}} |w(c_t, u)|}$.

The nearest neighbors algorithm complexity is in $o(k^2n)$ for k clusters and n users. Moreover, the whole complexity of the construction of the tree yields $o(n \log_2 n)$. The final center of each leaf of the FRAC tree corresponds to a profile of typical users. It means that we consider these centers as virtual users synthesizing the preferences of each subset of users.

The profiles of typical users are then sent on client side, using IP multicast addresses. Subsequently, the system uses the Pearson correlation coefficient to compute distances between the active user and the typical users. We consider that the active user belongs to the community whose center is the closest to him/her. At last, we can predict the interest of the active user for a resource r_l with the equation 3.

$$p_{u_a,r_l} = \max(r_{min}, \min(r_{u_t,l} + (\bar{r}_{u_a} - \bar{r}_{u_t}), r_{max}) \quad (3)$$

With: u_a the active user;
 u_t the nearest typical user;
 p_{u_a,r_l} the prediction of u_a for r_l ;
 \bar{r}_u the average of votes of u ;
 r_{min} and r_{max} respectively the min and max bounds of the vote scale.

The identification phase of the active user to one of cliques is in $o(2p)$ on client side, where p corresponds to depth of the tree. Even when the active user didn't want to share his/her preferences, it is possible to do predictions since they are made on client side.

The clusterization can be performed so that cliques hold about the same number of persons for a given depth of the tree. In this way, we introduce novelty in recommendations.

4 PERFORMANCE ANALYSIS

In this section, we compare our clustering algorithm with Item-item [10] and the Correlation-based Collaborative Filter CorrCF [9]. We have implemented all these methods in Java. The different tests have been done on a laptop computer equipped with a CPU 1.8 GHz and 1 Go of RAM. We evaluate these techniques in terms of computation time and relevancy of predictions.

4.1 Computation time

In order to compare the computation times of the aforementioned algorithms, we have generated matrices with different sizes. In this simulation, the votes of each user follow a Gaussian distribution centered on the middle of the representation space. We argue that this situation increases the number of iterations needed in the clustering algorithm, since the users are close to each other. Moreover, there is only 1% of missing data in the generated matrices. Consequently, we almost work in worse case for the computation time tests.

The results of these tests are shown in the table 1. The announced times include the writing of results in text files. The FRAC algorithm provides results in a quite short time. It is thus possible to apply it to large databases. For example, the system only needs about 6 or 7 minutes to compute typical behavior profiles with 10.000 users and 100 items. In the same case, the CorrCF algorithm requires several hours of computation (one of which is spent to compute the similarity matrix). The response time of CorrCF increases besides exponentially with the size of the database and is not in accordance with industrial constraints.

We note that Item-Item gets results much more quickly than FRAC when there is a lot of users and only few items. Nevertheless, the tendency may be reversed when the number of items grows, even if the number of users is still much more important (cf. table 1 for 10.000 users and 1000 items). Moreover, the corpus we used is not the most appropriate for our algorithm, since the number of items is almost twice as big as the number of users.

At last, we have tried to cluster huge populations. The FRAC algorithm was able to supply results in about 11 hours for 120.000 users and 150 items. This computation time is much smaller than the bouquet update period⁷. Consequently, it is completely suitable for the Casablanca software.

4.2 Recommendations relevancy

In order to compute the prediction relevancy in our system, we used the GroupLens database⁸. The latter is composed of 100.000 ratings of real users. Thus, we considered a matrix of 943 users and 1682 items. Moreover, each user has rated at least 20 items. The database has been divided into a training set (including 80% of all ratings) and a test set (20% of votes). We compare our algorithm with the three others by using the Mean Absolute Error (MAE).

MAE is a widely used metric which shows the deviation between predictions and real user-specified values. Consequently, we computed the average error between the predictions and the N ratings of the test set as shown in formula 4.

$$MAE = \frac{\sum_{i=1}^N |p_i - q_i|}{N} \quad (4)$$

The results are shown in the figure 4. The FRAC algorithm does predictions as good as the CorrCF – which is memory-based – and not so far from the Item-Item.

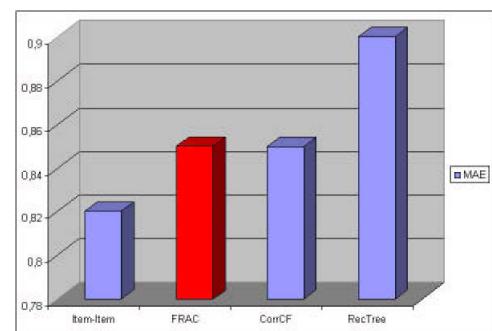


Figure 4. Comparison of prediction quality.

⁷ The bouquet is updated once a week. It means that the list of websites change. But the content of websites is updated every two hours.

⁸ <http://www.grouplens.org/>

Items Users	100			150			1000		
	FRAC	CorrCF	Item-Item	FRAC	CorrCF	Item-Item	FRAC	CorrCF	Item-Item
200	0"70	2"60	2'14	1"35	3"17	2"71	4"78	11"09	52"74
400	1"84	6"09	3"87	2"09	7"62	5"29	8"58	32"24	1'22"
600	3"60	11"78	5"59	4"10	15"21	7"34	15"43	1'04"	2'05"
800	7"03	19"98	7"23	7"34	25"67	10"53	30"17	1'52"	2'33"
1.000	8"30	30"22	8"56	9"45	40"68	12"84	39"71	3'06"	3'25"
1.400	11"21	1'00"	11"50	12"81	1'17"	18"10	49"47	6'04"	4'29"
10.000	6'50"	7h30'	1'22"	9'12"	-	2'05"	14'22"	-	49'28"

Table 1. Computation times of different collaborative filtering algorithms.

5 CONCLUSION

The novelty of our model relies on the fact that we have mixed a distributed collaborative filtering method with a behaviour modeling technique. The main advantage of this combination is to take into account in an overall way the strong constraints due to an industrial context such as the time constraints, the privacy, the data sparsity and the scalability to real commercial applications. Our recommender system has been implemented in a satellite broadcasting software with about 120.000 users. We also dealt with the ASTRA business plan by getting individual preferences – only stored on client side – in a transparent way.

Before our collaboration with ASTRA, Casablanca wasn't able to classify websites in order of preference. Thus, the filtering module was accepting the voted packages which were broadcasted until the cache was full. Most of the time, the content of the cache wasn't optimum and was even almost random. From now on, the system is able to anticipate the arrival of the packages and to decide if it is interesting to add them in the cache. The filtering module sorts out the websites and optimizes the suggestions according to the cache size defined by the active user. We are now evaluating the satisfaction of users as regards recommendations. ASTRA has recruited voluntary beta-testers to experiment the system in real conditions. We searched in the literature the different existing methodologies to measure the user satisfaction. We finally choose the approach in [1] because it is relying on an analysis of usage.

While waiting for the results of this long-term study, the off-line performance analysis highlights the fact that our algorithm gets rather quickly results on server side, even when the corpus is not very adequate. Moreover, the identification phase on client side is in $o(2p)$. This part of computations has been optimized in order to supply users with recommendations in a very short response time.

Another advantage of our algorithm is the stability of the model. Thanks to the new initialization phase, the results are reproducible when we launch the clustering process several times. Furthermore, contrary to some algorithms such as K-Means, the computations systematically end.

ACKNOWLEDGEMENTS

We want to thank ASTRA and the CRPHT which have encouraged this work.

REFERENCES

- [1] Anne Boyer and Regis Lhoste, 'Smart agent evaluation methodology (d4.1 addendum)', in *IST Elin Project N2000-30188*, (November 2004).
- [2] Paul S. Bradley and Usama M. Fayyad, 'Refining initial points for k-means clustering', in *Proceedings of the 15th International Conference on Machine Learning (ICML98)*, pp. 91–99, San Francisco, USA, (May 1998). Morgan Kaufmann.
- [3] John S. Breese, David Heckerman, and Carl Kadie, 'Empirical analysis of predictive algorithms for collaborative filtering', in *Proceedings of the fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 43–52, San Francisco, CA, (July 1998).
- [4] Sylvain Castagnos and Anne Boyer, 'From implicit to explicit data: A way to enhance privacy', in *Workshop on Privacy-Enhanced Personalization (CHI 2006)*, Montreal, Canada, (April 2006).
- [5] Philip Chan, 'A non-invasive learning approach to building web user profiles', in *Workshop on Web usage analysis and user profiling, Fifth International Conference on Knowledge Discovery and Data Mining*, (August 1999).
- [6] S. H. S. Chee, J. Han, and K. Wang, 'Rectree : An efficient collaborative filtering method', in *Proceedings 2001 Int. Conf. on Data Warehouse and Knowledge Discovery (DaWaK'01)*, volume 2114, pp. 141–151, Munich, Germany, (September 2001). Lecture Notes in Computer Science.
- [7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, 'An algorithmic framework for performing collaborative filtering', in *In Proceedings 1999 Conference of Research and Development in Information Retrieval*, pp. 230–237, Berkeley, CA, (August 1999).
- [8] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles, 'Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach', in *Proceedings of the sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pp. 473–480, San Francisco, USA, (2000). Morgan Kaufmann Publishers.
- [9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, 'Grouplens: An open architecture for collaborative filtering of netnews', in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175–186, Chapel Hill, North Carolina, (1994). ACM.
- [10] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl, 'Item-based collaborative filtering recommendation algorithms', in *World Wide Web*, pp. 285–295, (2001).
- [11] Upendra Shardanand and Patti Maes, 'Social information filtering: Algorithms for automating "word of mouth"', in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pp. 210–217, (1995).
- [12] L. Ungar and D. Foster, 'Clustering methods for collaborative filtering', in *Proceedings of the Workshop on Recommendation Systems*, Menlo Park California, (1998). AAAI Press.

Software Companion

The MEXAR2 Support to Space Mission Planners

Amedeo Cesta and Gabriella Cortellessa and Simone Fratini and Angelo Oddi and Nicola Policella¹

Abstract. This paper describes a fielded AI system in daily use at the European Space Agency (ESA-ESOC) since February 2005. The tool, named MEXAR2, provides continuous support to human mission planners in synthesizing plans for downlinking on-board memory data from the MARS EXPRESS spacecraft to Earth.

The introduction of the tool in the mission planning workflow significantly decreased the time spent in producing plans. Moreover MEXAR2 improves the quality of the produced plans thus guaranteeing a strong reliability in data return enabling a more intensive science activity on board. The introduction of MEXAR2 has modified the role of the human mission planners who can now evaluate and compare different solutions rather than dedicating their time exclusively to computing single solutions (a tedious and repetitive task which does not capitalize on the mission planners' decision-making expertise). These characteristics have effectively made MEXAR2 a fundamental work companion for the human mission planners.

1 CONTEXT

A critical problem for interplanetary space missions is maximizing science while guaranteeing data return to Earth. Additionally, the reduction of investments in the field, requires space programs to perform ambitious tasks with more limited budgets with respect to the past. Europe's MARS EXPRESS is seen, quoting the mission web description, "as a pilot project for new methods of funding and working". Specifically, an innovative working relationship between ESA, industry, national agencies and the scientific community, as well as the reuse of equipment developed for the ESA Rosetta mission, were important factors that have contributed to the development of MARS EXPRESS as a relatively low cost mission².

Notwithstanding the low budget, the mission exposes ambitious goals for the scientific experiments on board. The seven payloads the orbiter is equipped with, are expected to maximize their data return to take advantage of the opportunity offered by the proximity to the Red Planet. Indeed an amount of novel information from Mars is arriving to the space science community and, through the media, to citizens. We may remember either the accurate pictures taken by the High Resolution Stereo Camera (HRSC) which image the entire planet in full colour 3D and with a resolution of about 10 meters, or the information about the distribution of water, both liquid and solid, in the upper portion of the crust of Mars, distributed by MARSIS, the Mars Advanced Radar for Subsurface and Ionosphere Sounding.

Obviously, in a deep-space mission like MARS EXPRESS, data transmission to Earth represents the fundamental aspect. The space-probe continuously produces a large amount of data resulting from the activities of its payloads and from on-board device monitoring and verification tasks (the so-called *housekeeping* data). All these data are to be transferred to Earth during bounded downlink sessions.

Moreover, in the case of MARS EXPRESS, a single pointing system is present. This implies that, during regular operations, the space-probe either points to Mars, to performs payload operations, or points to Earth, to download the produced data. As a consequence, on-board data generally require to be first stored in a Solid State Mass Memory (SSMM) and then transferred to Earth.

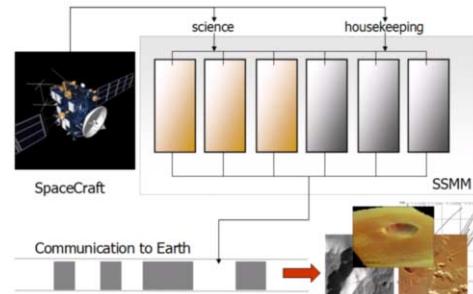


Figure 1. Bringing data from Mars to Earth

In Fig. 1 the main data flow is shown: the spacecraft instruments produce data that are stored in the SSMM on-board memory. The main problem to be solved consists in synthesizing sequences of spacecraft operations (*dump plans*) that are necessary to deliver the content of the on-board memory during the available downlink windows. The average data production on a single day can be around 2-3 Gbit while the transmission rate of the communication channel, which varies between 45Kbs and 182Kbs, may not be sufficient. The on-board memory is subdivided into different banks, called *packet stores*, in which both scientific (*science* from payloads) and spacecraft management data (*housekeeping*) can be stored. In particular housekeeping has to be guaranteed to arrive to Earth daily so as to allow checking for the safety of the different operations on board. Notice that each packet store assigned to science data is managed cyclically, so in case new data are produced before the previous are dumped to Earth, older data are overwritten and the related observation experiments have to be re-scheduled. Even if the on-board memory is about 9.4 Gbit the irregular distribution of transmission windows, the different transmission rates of such windows and the different data rates for data production (e.g., the stereo camera can produce files close to 1Gbit) may often create usage peaks close to the packet store capacities.

To complicate matters there is an additional uncertainty factor in data production for some instruments due to different compression algorithms. Usually dump plans for the on-board memory are computed for a nominal expected production of a certain payload activity, a POR (Payload Operation Request), but mission planners may discover from housekeeping check that data on-board are more than expected so they have to recompute the *dump plan*, i.e., the sequence of dump commands that implements the memory download policy.

Content. This paper describes how an AI-based interactive tool, called MEXAR2, has been inserted in the continuous loop of mission operation management, allowing the mission planners to gener-

¹ ISTC-CNR, Institute for Cognitive Science and Technology of the Italian National Research Council, email: name.surname@istc.cnr.it

² For details check the mission home page at <http://sci.esa.int/marsexpress/>

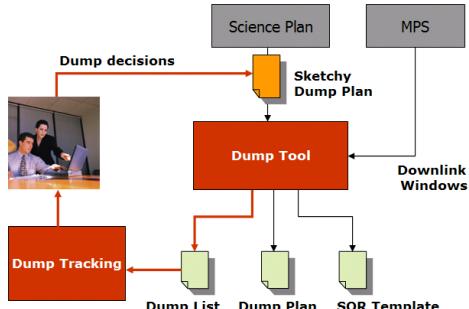


Figure 2. Dump Planning Approach before MEXAR2

ate dump plans in a shorter time, and to explore alternative solutions obtained by tuning a controlled set of parameters. MEXAR2 is in daily use at ESA-ESOC since February 2005 and has been further improved during almost a year of operational use. The key improvement introduced by MEXAR2 is the fact that an AI software tool has contributed with a new perspective to the use of supporting software for mission decision: from a common practice in which the software help is mainly for constraint checking and all the decisions are left to the human planner, to a practice in which the software has an active role in taking decision and becomes a “companion” for the human decision-maker offering a wider perspective on the problem to solve.

The paper sheds light on some key aspects that acted as enabling conditions for the MEXAR2 success story. The aim is to show how the AI background that generated MEXAR2 significantly contributed in the effectiveness of the technology infusion program.

2 THE “TRADITIONAL” MISSION PLANNING

Consolidated practice in space mission operations always involves a Scientific Control Center responsible for science activities of the various payloads (oversimplifying sketched in Fig. 2 as Science Plan), and an Operational Control Center in charge of detailed operations for the maintenance of the spacecraft and the return of scientific data. More in detail the Operational Control Centre is responsible for various activities like the uplink of set of commands, the downlink of science and housekeeping data, the monitoring of the spacecraft, the negotiation with the Science Control Center about possible adaptations and refinements of the master Science Plan. For the sake of simplicity we identify all these activities as Mission Planning System, MPS. One fundamental activity within MPS, consists of generating the Dump Plan, a sequence of Space Operation Requests (SORs) that ask the spacecraft to transmit a certain amount of data from a given packet store to Earth.

Fig. 2 depicts the work practice, within the MPS, that was established during MARS EXPRESS operation to generate dump plans, before the introduction of the MEXAR2 tool. A small team of people (mission planners), one of them being the main responsible for the task, received an initial Sketchy Dump Plan form the Scientific Control Center which, for each payload, listed the set of scheduled PORs and the relative data that were to be transferred to Earth. Starting from this initial Sketchy Dump Plan, the mission planners were in charge of refining it and producing three elaborations, namely the final Dump Plan, the Dump List and a SOR Template, a version of the dump plan formatted to be directly executed on-board.

Two simple software tools, helped the mission planners in their daily tasks, both slight enhancements of a spreadsheet file: (a) the Dump Tool that, taking as input the current Dump Plan (essentially a text file) and a specification of the downlink windows provided by MPS specialists, performed some constraint checking and generated the output formats for Dump Plan and SOR Template plus the additional file, called Dump List; (b) the Dump Tracking, a simple program that allowed easy inspection of the Dump List file. In particular the Dump List file includes information on both data storages

and data dumps distributed on the various packet stores of the SSMM, and allows to compute capacity profiles over time to check for over-production of data and, as a consequence, data loss (overwriting). The normal cycle of work consisted in the mission planners initializing the Sketchy Dump Plan with nominal data quantities extracted from the PORs in the Science Plan, using the two tools to create the current projections on the different packet stores according to the downlink windows, and analyzing the results of the Dump Tracking to refine, through updated dump decisions, the Sketchy Dump Plan. This loop was iteratively repeated until a satisfactory Dump Plan was computed.

Problems. This cycle is also common to different missions, but, as said before, MARS EXPRESS is strongly devoted to maximize science return, and the visibility windows are quite limited – other missions may allow time enough to regularly perform complete memory dumps. The previous procedure ended up being very time consuming because mission planners were essentially proceeding by “Trial and Error” on a textual plan, deciding to move back and forth in time the existing dump commands or splitting a current dump in multiple dumps allocated on different time windows. Two additional complicating factors were: (a) the activity of “manual” refinement was possible having a reference period of one or two days but was cognitively very difficult on a larger horizon; (c) the uncertainty factor on data production made things even worst because in certain cases, from housekeeping analysis, planners discovered that data still onboard from the previous day were more than expected and they had to restart the refinement process of the Dump Plan to take this additional data into consideration. In general this planning process too often produced dump plans not sufficiently reliable with respect to overwriting, and also entailed a significant amount of time spent on repetitive and tedious low level tasks.

Addressing problems with MEXAR2. The environment of space operation control center is the most conservative of all the work environments involved in a space program. This is to say that it is not the place where you can say “Hey! this is an AI tool for you to try”. People daily work at guaranteeing survival of the spacecraft and maximal robustness of its operations. The more tested before is a solution the more likely it will for it to be adopted. On this aspect, authors recognized they have been lucky, three years before operations, to have had the opportunity to study the memory dumping problem of MARS EXPRESS. This previous experience produced a working prototype, named MEXAR, able to solve a core problem close to the current one and allowed to start a trustful interaction with the MPS people. The prototype was not working on real data, because not available at that time, but different algorithms have been developed based on a CSP approach [5, 4]. The daily dump plan synthesis at ESA was so critic, and time consuming for humans, that in June 2004 we received a detailed description of the real data format and have been entrusted with the task of possibly improve the current practice on this work.

Thanks to the use of an AI declarative problem model we have been able to capture the data flow from real data very quickly (August 2004) and deliver a first solver able to synthesize dump plans even if several detailed constraints were still missing. Then a new solver has been developed, able to capture exactly the dump problem, and its solution have been validated by mission planners since October 2004. For six months our automated procedure have been continuously used back to back with the usual procedure in Fig. 2, and incrementally subtle bugs have been detected and solved. Since February 2005 our solver has replaced the previous procedure and ESA has asked us to design an interactive version of the tool and to further extend the capability of the solver. Since July 2005 an interactive AI tool, named MEXAR2 is entered in the daily loop of mission practice. A main advancement for the solver has been to solve multiple mission days problems while the interaction part offers to the mission planners a set of additional statistics and information on the

current solution and the possibility to obtain and compare different solutions.

3 AUTOMATING THE DUMP SYNTHESIS

The description of the details of MEXAR2 is quite synthetic due to space limit. This section and the next will show what kind of ideas has been integrated in the system.

Identifying relevant domain entities. The modeling activity to capture the basic need of the dump synthesis produces a symbolic model of the domain entities shown in Fig. 1. Our planning and scheduling background brought us to model the domain, identifying *resources* that serve *activities*. Two different types of resources are used for modeling (a) the different *packet stores*, pk_i of the SSMM. Each pk_i has its own fixed capacity, c_i , and priority, pri_i ; (b) the *communication channel*, that represents the downlink connections to Earth for transmitting data. This resource is characterized by a set of separated communication windows w_j that identify intervals of time in which downlink connections can be established. Each element w_j is a 3-tuple $\langle r_j, s_j, e_j \rangle$, where r_j is the available data rate during the time window w_j and s_j and e_j are respectively the start and the end-time of the window.

Activities a_i are characterized by a fixed duration d_i and two variables s_i and e_i which respectively represent their start- and end-time. Two types of activities are relevant, store operations st_i and memory dumps md_i . A store operation, st_i “instantaneously” stores, at its end-time e_i , an amount of data q_i in a defined packet store, pk_i . Two different types of data productions can be modeled by using store operations st_i : the *Payload Operation Request* (POR) from science, over certain intervals, and the constant rate Housekeeping Production.

Through a memory dump, $md_i = \langle pk_i, s_i, e_i, q_i \rangle$, an amount q_i of stored data is transmitted from the packet store pk_i to the ground station, during the interval $[s_i, e_i]$.

Problem constraints. A MEX-MDP (Mars Express Memory Dumping Problem) problem is composed of a set of scientific observations and a set of housekeeping productions allocated on a time horizon $\mathcal{H} = [0, H]$.

A *solution* (Dump Plan) consists of a set of dumping commands $S = \{md_1, md_2, \dots, md_{nd}\}$ such that a number of constraints are satisfied. Among such constraints there are: (a) at each instant of time $t \in \mathcal{H}$, the amount of data stored in each packet store pk_i does not exceed the packet store capacity c_i , i.e., overwriting is not allowed; (b) the whole set of data is “available” on ground within the considered temporal horizon $\mathcal{H} = [0, H]$, except an amount of residual data for each packet store pk_i less or equal to the capacity c_i ; (c) Each dump activity, md_i , is executed within an assigned time window w_j which has a constant data rate r_j . Moreover, dump commands cannot mutually overlap.

During the mission practice several decisions have been introduced as additional constraints for the problem solving. In particular, one of them strongly influences the problem solving: the *housekeeping data* describing the spacecraft internal status, need to be analyzed immediately by the MPS and consequently are to be downloaded within each operational day. Additionally, while the science packet stores can be preempted (multiple dumps are allowed), the housekeeping packet stores should be dumped with a single command. This specific constraint required us to create a hybrid algorithm that first performs a search to allocate housekeeping, then synthesizes the different science dumps on the remaining downlink window capacities.

Further requirements from the mission planners concerned the specification of priorities between different packet stores.

Additionally the mission planners had to minimize effects due to the uncertain estimation of data produced by some payloads. With respect to this we have investigated the issue of *solutions robustness*.

Algorithm 1: Dump Plan Generation

```

Input: problem P, policy parameters robust and priority
Output: Sequence of down-link commands Cmd
// Data Dump Level
// Produce a data allocation over the
// communication channel
while S uncomplete do
    S ← HousekeepingAllocation (P, priority)
    S ← MaxFlowAllocation (P, S, robust, priority)
    if No more HK allocation then
        break
// Packet Level
// Generate data commands
Cmd ← GenerateDownLink (S, priority)
return Cmd

```

Here robustness is oriented toward controlling the level of memory use in order to avoid possible loss of data due to overwriting. A possibility to overwrite data may occur when a volume of data greater than expected has to be stored and there is not enough space in the packet store. For this reason we define as robust, the solution in which a specified amount of space is preserved on each packet store in order to safeguard against overwriting. In other words, solution’s robustness is related to the concept of *distance to the overwriting state*. Hence, we consider the peaks of data in a packet store close to its maximum capacity as sources of schedule’s brittleness. A possible way to increase solution’s robustness is to *flat* these peaks by finding a different distribution of the dumping operations within the horizon $[0, H]$.

An effective hybrid solver. Given the domain entities of a MEX-MDP instance, two levels of abstraction for the problem domain have been considered: (a) a first level, *Data Dump level*, through which the amount of data to dump from each packet store for each time window, is assessed; (b) a second level, *Packet level*, that, given the results of the *Data Dump level*, generates the final dump commands. Such a problem *decomposition* is motivated by the complexity of the problem. In fact, this abstraction allows us to focus on the *dominant* aspects of the problem, i.e. data quantities, packets store capacities, and dump capability over the communication links, without considering the problem of generating dump commands. It is worth remarking that the packet level step can be automatically accomplished once a solution for the Data Dump level is computed (no failure is possible) so our main focus has been on the Data Dump level.

A high level description of the solving process is shown in Algorithm 1. The solving process contains two steps: a first which produces a data allocation over the communication channel. In particular, for each packet store and for each time window, the amount of data to download is selected. The second step, generates the down-link commands starting from the previous allocation. As said before the key aspect is the generation of the data dumps. This step is represented in Algorithm 1 by the *while* loop. Each iteration of the loop calls the following two procedures:

- *HousekeepingAllocation*: this procedure aims to find a consistent solution of the sub-set of the housekeeping packet stores. In fact these packet stores contain more important data (for spacecraft safety), no residual data are allowed, and all the data (of a single packet store) have to be downloaded by using one command (no-preemption).
- *MaxFlowAllocation*: given a partial solution from the previous step, this procedure generates the data dumps from the remaining science packet stores. This procedure is based on a reduction of the problem P (plus the solution of the previous step, S) to a Max-Flow problem: the former problem has a solution when the maximum flow in the latter equates the total amount of data to dump [4].

It is worth remarking that the max-flow approach is used only on the set of scientific packet stores and not on the housekeeping ones. In fact the last procedure is not able to stop preemption. To do this the HousekeepingAllocation is a backtracking search that generates at each step a possible dump plan for the housekeeping packet stores. These two steps will be iterated until a complete solution is found or no further alternative exists (we thus check if there is any allocation available). As we are facing a real problem we need to always give an answer. Then, at the end of the while loop we will have a solution S that is used to generate the final list of dump commands.

Additionally for all the procedures it is possible to set different parameters. The allocation in HousekeepingAllocation can be done considering the priority values of the housekeeping packet stores (parameter **priority**). The procedure MaxFlowAllocation accepts also a parameter **robust**. In this case we have an iterative max-flow based search that tries to flat possible peaks in the usage of the storage devices [4]. Also GenerateDownLink can consider the packet store priority values. In fact the input of this procedure consists of a set of data dumps for each time window. These are allocated in the interval but are not ordered, therefore they are ordered according to the priority values.

4 INSERTING MEXAR2 IN WORK PRACTICE

Besides the problem solving ability in synthesizing memory dumps, two other factors have been instrumental to the MEXAR2 adoption in work practice: (a) the seamless integration of the tool in the data flow of the consolidated work practice. Being the mission operational since more than six months, the fact that no change has been required in the format of data and in the current habit apart the fact to have a further support to the solution identification, has been very important; (b) the fact that the tool allowed a mixed initiative interaction style since its very early release. In MEXAR2 the users may interact with the tool, influence its behavior, and generate alternative plans, either changing the model or tuning the algorithm. This aspect has been a key factor for fostering users' contribution in a joint work with the tool, and for exploiting their knowledge, coming from experience and not easy to encode in the software.

These two aspects, have been instrumental to create users willingness to use the tool and ask for its modifications and improvements. During this phase several user's constraints have been made explicit, and once inserted in the tool, ended up adding value to the mixed-initiative style and to its acceptance by users. From our standpoint, it has been very important to keep the iterations for the refinements fast enough. To this aim the underlying declarative symbolic AI model has been very important.

4.1 Capturing the mission data flow

The current procedure for synthesizing dump plans with MEXAR2 in the loop is shown in Fig. 3. We first observe that MEXAR2 accepts as input directly the POR requests from the science plan, and the specification of the downlink windows from MPS. It produces as output the dump plan in the three formats expected by ESA people. This has been obtained by encapsulating the solver between two software modules: a first one responsible for processing the input files and selecting the relevant information for the symbolic model used by the solver, and a second one for generating the output according to external formats.

A second observation concerns the change of perspective offered to the mission planners. Users do not directly iterate in the attempt of "almost manually" producing the dump plan but rather establish a dialogue with MEXAR2, having access to the model of the domain and dials to tune the algorithms. From MEXAR2 they receive also additional information, for short referred to as Statistics in Fig. 3. This information enrich their ability to analyze the current solution. In general we have pursued the goal of allowing users to take more

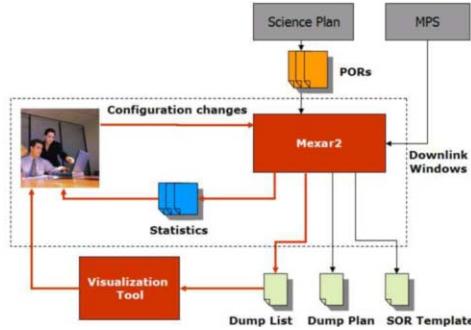


Figure 3. Mixed-Initiative Dump Plan Synthesis

strategic decisions, and maintain control over the synthesis of SORs, delegating to MEXAR2 not only the repetitive part of the work but also the proactive role of creating the dump plan. This has been achieved by exploiting MEXAR2 ability to work quickly and to consider clues given by solving strategies. Additionally MEXAR2 allows now to consider a temporal horizon previously impossible to tackle.

4.2 Fostering mixed-initiative problem solving

In Fig. 3 a dashed box underscores the main human-computer interaction loop fostered between users and MEXAR2. Given a specific dump period and an initial state of the world, the proactive role of MEXAR2 consists in computing a feasible dump plan for the period. Being relieved from the tedious and difficult part of the task, the mission planners can now concentrate on strategic decisions. MEXAR2 can be in fact iteratively configured to reproduce changes in the domain or to tune the parameters of the solving algorithms. In particular it is possible to create alternative configurations of the tool which correspond to diverse situations of the real world (e.g., different level of residual data, extra dumps based on incidental events, changes in priority of data to be downloaded, etc.) and quickly obtain the corresponding solutions from the tool.

Additionally users can tune the solver parameters (e.g., to take into consideration dangerous peaks of the packet stores level) thus actively contributing to the problem solving and guiding the search toward more intelligent and robust solutions.

A further valuable help of the mixed-initiative style is the possibility to jointly build, save and evaluate several solutions with the aim of avoiding potential data loss and more in general to look for more optimal results. Indeed, by using MEXAR2 as an active collaborator, overwrites can be quickly detected during the medium term planning process and fed back to the science community for POR update (i.e., to correct the case of anticipated data loss).

The main interaction module of MEXAR2 (see Fig. 4), reflects immediately the dialogue with the user: (a) the channel from mission planners to MEXAR2 is implemented through the pane called "Input Actions"; (b) the communication from MEXAR2 to the users is implemented mainly through the pane "Results and Statistics" that provides different output and secondarily through the pane "MEXAR2 Messages" that provides messages during each computation.

Input Actions. The basic action consists in the specification of the dump interval, the starting and ending dumping days – e.g, Starting Day 156, 2005; Ending Day 171, 2005 in Fig. 4.

Then, through three sub-panels, users communicate to MEXAR2 additional information. In particular (a) the "*configure*" action allows to specify the domain model, e.g., the specification of packet stores, their size, their respective priorities, their need to be robustified, etc.; (b) the "*initialize*" action communicates to MEXAR2 the residual memory stores from the previous temporal interval that are to be downloaded together with data related to new PORs; (c) "*selecting*" through this pane users can tune the parameters of the algorithms.

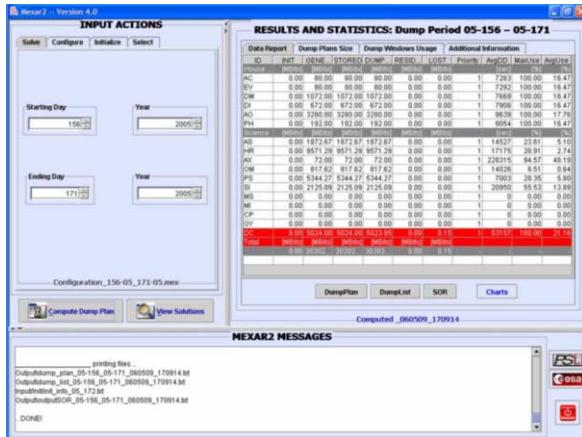


Figure 4. MEXAR2 Interaction Module

Automated solution. Once the needed context and the required parameters have been specified, users may ask the solver to compute a dump plan by clicking the “Compute Dump Plan” button. Feedback about the solver decisions is provided through the MEXAR2 Messages pane. As mentioned MEXAR2 provides different solving algorithms. A basic solving procedure looks for solutions without considering the packet stores priorities. A second one, takes into account the priorities, while a third algorithm aims at obtaining robust solutions. The pursued idea of robustness is the one of avoiding that a single memory bank is extremely full of data so that any variation at run time is difficult to be absorbed with consequent high risk of overwriting. An additional aspect that the algorithms have addressed concerns the “fragmentation” of the whole dump plan. In this respect the user can specify different thresholds, called input and output thresholds, that avoid production of commands too short or that dump too little data.

Inspection of Results. Once the solver has found a solution, the Interaction Module shows results exactly as in Fig. 4. A set of statistics are shown in the “Results and Statistics” pane that provides aggregate information on the current solution. In particular a panel named “Data Report” is shown, with entries subdivided into packet stores. This report gives the user an immediate view of the input/output data balance. Apart the Data Report table, the basic pane allows to access the three main files that are produced by the solver through three buttons (“Dump Plan”, “Dump List”, “Output SOR”). Recently ESA provided us a Visualization tool, developed in-house to validate results of MEXAR2, see Fig. 3, that has been inserted as an additional plug-in.

Computing different solutions. As said before we have worked to endow the system with functionalities that support the user in exploring different solutions. The mission planner can indeed generate different *use profiles* of the tools acting on the “configure” and “select” features and compare various results. To facilitate this activity we have also introduced an additional structure called *Solution Data Base* that, for any time interval, shows a table with all the solutions computed for the period. For each solution in the table, users can see the settings used to obtain it and a summarization of different metrics for evaluation (further details are omitted for lack of space).

5 SUMMARY OF ACHIEVEMENTS

In evaluating the main results obtained with MEXAR2 we consider two perspectives, namely the advantages for mission planners and the advantages for science management.

With respect to MPS a considerable advantage stems in the mixed-initiative style of the tool. The role of software which supported the

decision making in previous practice was really secondary and relegated to constraint checking tasks. On the contrary MEXAR2 is proactive with respect to plan synthesis while fosters, at the same time, human intervention and control during problem solving. In general data dump generation can now be performed quicker and with less effort and the quality of plans exceed that provided by the tools previously used. The problem of uncertainty in data production is addressed very satisfactorily from the user standpoint and the time saved for producing plans has been estimated to 50%.

The ability of MEXAR2 to generate plans over multiple days very quickly, allows mission planners to consider alternative solutions in order to avoid data loss. It is worth noting that before the introduction of MEXAR2 in the Mission Planning System, mission planners were satisfied with a single solution, while the tool guarantees the possibility of interesting optimizations. In fact, this allows to identify problematic intervals in the near future and negotiate with scientists alternative allocation of involved Payload Operation Requests thus minimizing overwrites.

The science community benefits from MEXAR2 as well: data from observations are available earlier and loss of data minimized. As already said potential overwrites can be quickly detected and fed back to the science community for PORs update. Thanks to MEXAR2 the use of the downlink channel is optimised and more data can be down-linked, thus allowing increase of science return from the mission.

6 CONCLUSIONS

AI planning and scheduling technology have been operationally used in US space missions from NASA and JPL [3, 6, 1, 2]. This paper has presented MEXAR2 the first case of operational use of such technology within ESA programs. We have described how a key aspect for the success have been the seamless integration of the tool in the work practice and the choice of developing an interacting software companion more than an isolated solver. Enabling conditions have been the fact that the dump plan synthesis problem was very critical during actual mission operations and that the very initial version of MEXAR2 have been soon able to cope with the real data flow at MPS.

Acknowledgements. This work has been sponsored by the European Space Agency (ESA-ESOC) under contract No. 18893/05/D/HK(SC). The authors would like to thank the MARS EXPRESS mission control team at ESA-ESOC for the valuable feedback received and in particular Erhard Rabenau and Alessandro Donati for their continuous assistance.

REFERENCES

- [1] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.C. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B.G. Chafin, W.C. Dias, and P.F. Maldague, ‘MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission’, *IEEE Intelligent Systems*, **19**, 8–12, (2004).
- [2] S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, D. Mandl, S. Frye, S. Shulman, J. Jones, and S. Grosvenor, ‘An Autonomous Earth-Observing Sensorweb’, *IEEE Intelligent Systems*, **20**, 16–24, (2005).
- [3] A.K. Jonsson, P.H. Morris, N. Muscettola, K. Rajan, and B. Smith, ‘Planning in Interplanetary Space: Theory and Practice’, in *Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)*, (2000).
- [4] A. Oddi and N. Policella, ‘A Max-Flow Approach for Improving Robustness in a Spacecraft Downlink Schedule’, in *Proceedings of the 4th International Workshop on Planning and Scheduling for Space, IWPSS’04*, ESA-ESOC, Darmstadt, Germany, (June 23–25 2004).
- [5] A. Oddi, N. Policella, A. Cesta, and G. Cortellessa, ‘Generating High Quality Schedules for a Spacecraft Memory Downlink Problem’, *Lecture Notes in Computer Science, LNCS 2833*, (2003).
- [6] B.D. Smith, B.E. Engelhardt, and D.H. Mutz, ‘The RADARSAT-MAMM Automated Mission Planner’, *AI Magazine*, **23**(2), 25–36, (2002).

ECUE: A Spam Filter that Uses Machine Learning to Track Concept Drift

Sarah Jane Delany¹ and Pádraig Cunningham² and Barry Smyth³

Abstract.

While text classification has been identified for some time as a promising application area for Artificial Intelligence, so far few deployed applications have been described. In this paper we present a spam filtering system that uses example-based machine learning techniques to train a classifier from examples of spam and legitimate email. This approach has the advantage that it can personalise to the specifics of the user's filtering preferences. This classifier can also automatically adjust over time to account for the changing nature of spam (and indeed changes in the profile of legitimate email). A significant software engineering challenge in developing this system was to ensure that it could interoperate with existing email systems to allow easy management of the training data over time. This system has been deployed and evaluated over an extended period and the results of this evaluation are presented here.

1 INTRODUCTION

Spam email has proved to be a problem that is enduring and difficult to solve. In January 2004 Bill Gates predicted that spam email would be eradicated as a problem within two years⁴. The fact that this prediction did not come to pass demonstrates the severity of the problem. Spam is difficult to prevent because of the very open nature of electronic email and because the cost of sending email is close to zero. So even if the rate of return from spam is very small (less than a fraction of a percent) the practice is still worthwhile and there is a constant *arms race* between spammers and email system administrators as each moves to circumvent the initiatives of the other.

Of the wide range of strategies that have been employed to combat spam some of the more effective have been; whitelists and blacklists⁵, authentication based techniques⁶, collaborative filters[10] and content-based filters. In this paper we describe a system called ECUE (Email Classification Using Examples) that belongs to the category of content-based filters. The objective in the design of ECUE has been to produce a filter that learns from examples and can update itself over time to handle the changing nature of spam. ECUE is a fully engineered system that has been deployed in trials over an extended period with a number of users. ECUE interoperates with, but is independent of, the mail user agent (MUA) of these users and the

trials show that it is effective at tracking the changing nature of spam (see section 4).

Since the focus in ECUE has been on handling *concept drift* in email, the two main challenges in the development of the system have been to select an effective case-base from the volume of training data available and to update this case-base over time [7]. The issue of managing the volume of training data requires a case-base editing policy that selects those training examples that are better at prediction than others. This case-base editing technique is called Competence Based Editing (CBE) and has two stages, a noise reduction phase called Blame Based Noise Reduction (BBNR) and a redundancy elimination phase called Conservative Redundancy Reduction (CRR) [5].

The case-base update policy that handles concept drift centers on two hierarchical levels of learning. The first and simplest level is a continuous case-base update with training examples that have been misclassified by our filter. The second level is a periodic retraining of the classifier to reselect features that may be more predictive of spam and legitimate email. We will show how this update policy combined with the initial CBE case-editing procedure can effectively handle the concept drift that is so evident in the spam filtering domain.

In addition to these research challenges, the deployment of ECUE involved the software engineering challenge of integrating the filter with the users' MUA. The overall architecture of the system is described in detail in section 3 and an assessment of the performance of the system in filtering spam is described in section 4. Before that, a review of other work on using machine learning techniques for spam filtering is described in the next section.

2 REVIEW

Research into the use of machine learning techniques for building spam classifiers fall into two categories, those that are evaluated on static datasets in offline environments and those that are evaluated in online, real-time environments. The majority of the research falls into the former category [2, 4, 8, 9, 11, 13, 18, 19] with little published research showing how effective these techniques are at actually filtering real email over time. There are two key machine learning systems that have been evaluated against live email; Filtron [15] and Spamat [1].

Filtron is a prototype anti-spam filter that was designed based on a comprehensive off-line empirical evaluation of four learning algorithms, Naïve Bayes, Flexible Bayes, LogitBoost and Support Vector Machines (SVMs) [3]. It is a Java implementation that runs on Unix platforms only and was evaluated by a single user over seven months. The classifier used was an SVM as that showed the best performance in the offline evaluation, although the system is configurable and dif-

¹ Dublin Institute of Technology, Kevin St., Dublin 8, Ireland, email:sarahjane.delany@comp.dit.ie

² Trinity College Dublin, Dublin 2, Ireland, email: padraig.cunningham@cs.tcd.ie

³ University College Dublin, Dublin 4, Ireland, email: barry.smyth@cs.ucd.ie

⁴ http://www.theregister.com/2004/01/26/well_kill_spam_in_two/

⁵ www.email-policy.com/Spam-black-lists.htm

⁶ www.emailauthentication.org/

ferent classifiers can be selected. The system was initially trained on 2313 legitimate emails received by the user and 1826 general spam messages and used 520 features. It was never retrained over the period it was used and the performance was very good with 52 False Positives (legitimate emails incorrectly classified as spam) reported out of 5109 legitimate mails received (approx 1.0%) and 173 out of 1623 spam received missed by the filter (10.6%).

Albrecht et al.'s Spamat filter [1] is an open extendable spam filter framework implemented in Java utilising a plug-in filter architecture. The author's initial beta-test evaluation used a variety of plug-in filters including a number of URL-based filters, a Naïve Bayes classifier, a rule-based filter and a collaborative filter. This evaluation which filtered approximately 90,000 messages from a dozen users, resulted in 0.5% False Positives and 7% spam emails that were missed by the filter.

3 ECUE

A key requirement of our spam filtering system is that it integrates with or works alongside the MUA or mail reader rather than replacing it. This allows the user to continue to use the mail reader software with which they are familiar. To this end, the system architecture has been designed to support initially the Internet Message Access Protocol (IMAP) protocol [12]. One advantage of IMAP over POP3 (the other mail protocol) is that IMAP supports the storing of messages on the central server for access from multiple sites. By using IMAP to access the mailbox, messages can be filtered and flagged on the server and this allows the user to use any client mail reader application that supports IMAP to access and read their email. All the popular mail reader applications including MS Outlook, Mozilla, Netscape and Thunderbird support IMAP.

The user and the spam filtering system have to be able to interact for two reasons. Firstly the filter has to let the user know of emails categorised as spam and secondly the user has to be able to alert the filter to emails have been classified incorrectly. Since a requirement of the system is to integrate with existing mail readers rather than replace them, it is important to define a way of interacting with the user that is consistent across mail readers.

ECUE uses the IMAP mail folders as the means of system-user interaction. The filter places any emails it categorises as spam into a user-defined spam folder. It leaves any email that it classifies as non-spam in the Inbox. If the user finds any mails classified incorrectly they indicate this to the system by moving the emails from the folders they were in to or from the spam folder. So a False Positive (FP) email should be moved from the spam folder (where the filter had placed it) into any other folder, indicating that it should not be in the spam folder. Similarly a False Negative (FN) email, a spam email missed by the filter, should be moved to the spam folder to indicate that it should have been classified as spam. With this model for system-user interaction, existing mail clients do not have to be extended to enable them to be used with the spam filtering system. All interaction is at the mail server level. This is also a familiar spam interaction model for users. Figure 1 depicts the state transition diagram for an email message which shows all the possible states for an email message.

The main drawback of this process of user interaction is the requirement to monitor the spam folder for false positives. To address this ECUE produces a measure of classification confidence that can be used to partition messages classified as spam into definitely-spam and maybe-spam categories. The definitely-spam category need not be monitored for FPs [6].

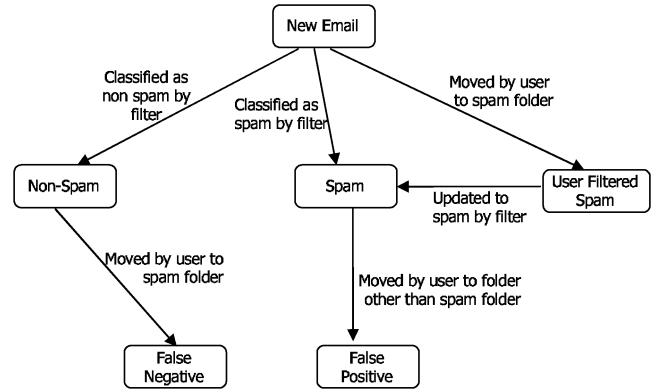


Figure 1. State Transition Diagram for an email message

In order to track the email message as it arrives and is filtered, an ECUE application specific header field is added to the email message. The value of this header field, which represents the state of the message as described in Figure 1, in conjunction with the folder the message is in indicates if and how the email has been moved by the user.

Mail folders are also used to identify the initial training data for the ECUE system. The user identifies the training emails which are to be used for the initial case-base set up by placing examples of their spam and legitimate emails into separate 'training' mail folders. These folders are identified in ECUE's configuration file and all emails in these folders are taken to be the initial training data.

3.1 The ECUE Learning System

The architecture of the learning system is described in Figure 2. The system uses previous examples of both spam and legitimate email received by the user as training data. In the initial training phase, the first process that the emails undergo is Feature Extraction which involves parsing or tokenising the text content of the training emails into features. No stop-word removal or stemming is performed on the text. Email attachments are removed before parsing but any HTML text present in the email is included in the tokenisation. As ECUE is a personalised filter, the header fields may contain useful information and a selection of header fields, including the *Subject*, *To* and *From* headers are included in the tokenisation. This idea is supported by a number of researchers [8, 17, 19] who concluded that the information from the message header is as important as the message body.

Three types of features are extracted; word features, letter features and statistical or structural features. The feature extraction process results in a large number of features for each training email. In addition, the representation of each email will be sparse, with only a small number of the total feature set having a value greater than zero. The Feature Extraction process for a typical training corpus will produce some tens of thousands of features. The task of Feature Selection is to identify which of these features are most predictive of spam or legitimate mails. The technique used for feature selection is Information Gain [16]. The output of feature selection is a reduced feature set where each training example email has a reduced set of feature-value pairs, including only those features identified as the most predictive. The number of features used by ECUE is configurable.

The task of Case Selection is to apply the Competence-Based Edit-

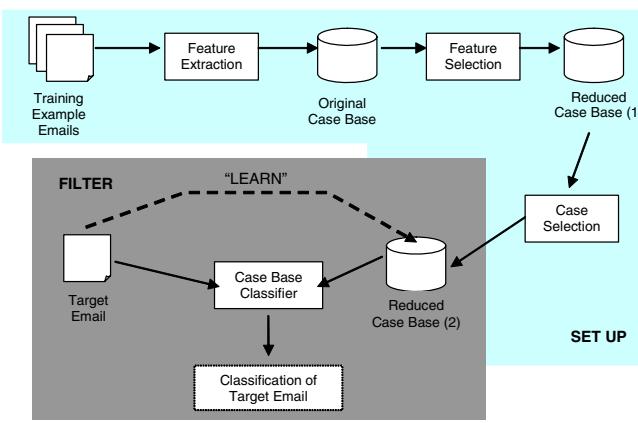


Figure 2. ECUE application structure

ing technique [5] which uses the competence properties of the examples in the case-base to remove noisy and redundant cases from the case-base. In effect Case Selection reduces the size of the case-base while ensuring that the generalisation accuracy of the system is not adversely affected.

In an example-based learner such as ECUE, the training examples are represented as cases in a case-base. Each training example is a case e_i represented as a vector of feature values, $e_i = (f_{1j}, f_{2j}, \dots, f_{nj}, s)$. The classification of new (or target) emails is performed using the k -Nearest Neighbour algorithm. As the case representation is binary, i.e. if the feature exists in the email the feature value $f_{ij} = 1$ otherwise $f_{ij} = 0$, a Case Retrieval Net [14] was implemented to speed up the retrieval process. The value of k is configurable and set up in the configuration file. A value of $k = 3$ was used for all evaluations presented in this paper. Due to the significance of FPs the classification process uses unanimous voting to bias the classifier away from such FP classifications. ECUE's unanimous voting requires all k neighbours retrieved by the Nearest Neighbour algorithm to be of class *spam* before the target case can be classified as spam.

ECUE uses different training data depending on the case-base that has to be built. If it is the first time the application is run after installation, ECUE uses training data that is placed by the user into two training folders in their mailbox. If a case-base is required to be built at any other time, e.g. when a feature reselection occurs, ECUE uses the most recent emails received by the user as training data. In these circumstances a percentage of the training data is made up of a selection of the most recently misclassified emails. The total number of emails to be used as training data is configurable as is the proportion of the training data that should comprise recently misclassified emails. This percentage is made up of all the FP emails previously identified (this number will be small) and an appropriate number of the previously identified FNs, randomly selected. An appropriate number of most recently correctly classified spam and non spam are then added to bring the training set up to the specified size.

When the user identifies emails that have been incorrectly classified by the system learning should take place. There are two levels of learning built into the system [7]:

- (i) incorrectly classified emails with their appropriate classification are regularly added to the current case-base;
- (ii) a feature re-selection process and a case-base rebuild is periodi-

cally performed on more recently received emails.

In order to help reduce and eliminate false positives, the system includes simple whitelisting which operates at two levels: Firstly, the user can define, in the configuration file, domains that will be acceptable to the filter. Any email that comes from these domains will be considered as legitimate. Secondly, the sender of all legitimate emails are maintained on a list and for all emails a case feature is set that indicates whether the sender is on the whitelist or not. This feature is used in the case-base retrieval process when identifying the most similar neighbours of the new email.

4 EVALUATION

ECUE was evaluated in a real-time online setting. The aim of the evaluation was to install the example-based spam filter in a 'live' environment and evaluate its performance. Since ECUE was designed to handle the concept drift in email the specific objective of the evaluation was to assess the *online* learning capabilities of the system. Over the evaluation period records were maintained of how ECUE performed both with and without the learning facilities.

ECUE was installed on the PCs of a number of users within the Computer Science department in two third level institutions in Dublin. The users were lecturers, postgraduate students and researchers within the departments. Since both institutions run a gateway spam filter (SpamAssassin) some users was asked to turn off SpamAssassin and to use ECUE for filtering their email. Others used ECUE as a second-level spam defense, filtering email that had passed through the gateway filter first.

When users identified emails that were misclassified by ECUE, they were asked to move the emails to the appropriate mail folders. Users were asked to initiate a feature reselection process when they felt that the performance of the filter was deteriorating or at least every couple of weeks.

The evaluation metrics used include:

- (i) The error rate; the overall proportion of emails that ECUE did not filter correctly (labeled %Error in all diagrams)
- (ii) The FN rate; the proportion of spam emails that ECUE missed (labeled %FNs).
- (iii) The FP rate; the proportion of legitimate emails that ECUE classified as spam (labeled %FPs).

For all these measures, figures are included for how ECUE performed when it attempted to handle the concept drift (i.e. with learning capabilites), labelled *with update*, and when the ECUE simply used the initial training data in filtering and did not attempt to handle the concept drift, labelled *no update*.

4.1 Evaluation Results

The evaluation was split into two phases. A preliminary evaluation involved an initial version of ECUE which included the case-base update facility, the first level of learning. The main online evaluation included both levels of learning, the regular update capability and the periodic feature reselection capability.

4.1.1 Preliminary Evaluation

Table 1 presents the results of the preliminary evaluation for six users. In addition to the performance figures, the table lists for each user the number of days that ECUE filtered the user's email and the

number of spam and legitimate emails that were filtered during that time period.

Table 1. Preliminary evaluation results

User	1	2	3	4	5	6
#days	28	34	30	50	70	41
Emails	#spam	969	618	80	890	3053
Filtered	#legit	191	422	390	2110	747
% Error	No update	8.1	9.6	7.5	20.7	22.0
	With update	5.6	6.5	4.5	8.0	10.7
% FPs	No update	1.6	2.4	5.0	0.6	1.5
	With update	1.0	2.1	1.8	0.7	2.7
% FNs	No update	9.4	14.5	20.0	68.3	27.0
	With update	6.5	9.5	17.5	25.4	12.6
						18.2

Analysis of these results show that ECUE performed better in all cases when update was enabled. Figure 3 shows a graph of the performance of ECUE for User 4. The graph shows the accumulated error (y-axis) over a certain number of emails (x-axis).

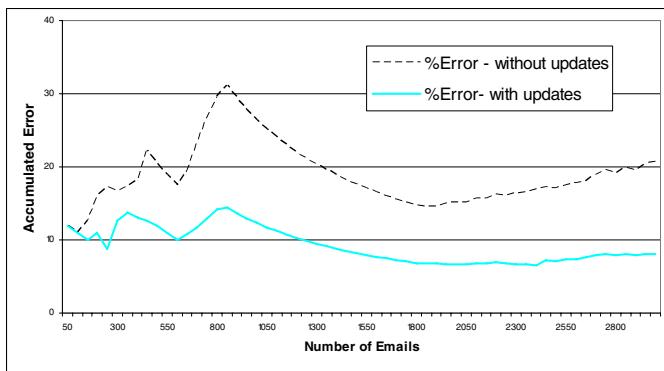


Figure 3. Performance of ECUE for User 4 in the preliminary evaluation.

However, improvements in the FP rate are not as consistent as the improvements in the FN rate. Four of the six users show improvements in the FP rate. Of the two remaining, one user (user 5) shows a considerable increase in FP rate from 1.5% without updating to 2.7% with updates. This may be explained by the fact that this user received very high levels of spam and the evaluation ran for a long period (70 days). As ECUE was very successful in handling the concept drift in the spam emails (the FN rate dropping from 27% to 12.6%) the system is being updated with a large number of spam emails to cope with this change and as a result becomes biased toward predicting spam.

The preliminary evaluation also showed that ECUE did not perform as well for users who receive high numbers of spam emails (users 5 and 6). These users had less than 90% of their email classified correctly whereas all other users had 92% or higher classified correctly. This indicated that it is necessary to include a further level of learning to allow a feature reselection to be performed to better handle the concept drift in the spam emails.

4.1.2 Full Evaluation

Table 2 displays the results of the full evaluation of ECUE. For each user the table lists the start and end date that ECUE ran on the user's

PC and the number of spam and legitimate emails that were filtered during that time period. The table also lists information about the training data used - the number of emails used (*initial size*) and what proportion of the training data was spam email (*%spam*). The number of times the user initiated the feature reselect process during the evaluation period (*#ftr reselects*) is included along with the performance achieved both with and without update, when update included a feature reselection process.

Table 2. ECUE full evaluation results

User	1	2	3	4
Filter Period	Start date	18-11-04	9-3-05	20-4-04
	End date	15-07-05	15-07-05	16-10-05
Emails	#spam	3689	4081	742
Filtered	#legit	1161	469	1480
Casebase	Initial size	308	299	201
	%spam	56%	54%	79%
	#ftr reselects	3	3	1
% Error	No update	32.8	21.8	17.3
	With update	6.1	4.7	12.1
% FPs	No update	0.3	0.0	1.3
	With update	0.7	0.2	1.1
% FNs	No update	43.2	24.4	49.3
	With update	7.8	5.2	34.0

Analysis of these results also show that ECUE performed better with update in all cases. Figure 4 shows a graph of the performance of ECUE for user 1.

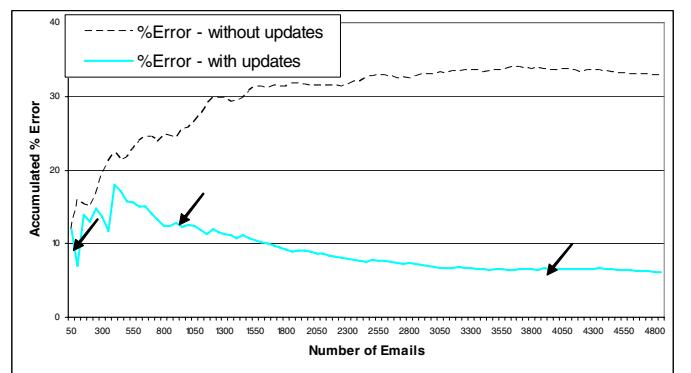


Figure 4. Performance of ECUE for User 1 in the full evaluation. The arrows show when the feature re-selection process occurred.

ECUE learned to recognise new examples of spam over the time period as is evident from the reduction in the FN rate for all users except user 4. The FN rate of user 3 did not drop as significantly as that of the others. This may be accounted for by the fact that all email received by user 3 was subject to initial organisation-level spam filtering on the mail server before it was forwarded to user 3's personal mailbox. As such, the spam email received by user 3 was spam that had been missed by the organisation-level filter and is possibly more difficult to recognise as spam. ECUE still identified 66% of user 3's spam correctly.

The bias of the classifier appears to be influenced to a large extent by the proportions of spam and legitimate email in the training data. User 4 received more than 12 times the amount of legitimate email as spam (possibly due to the success of the organisation level filter to

which user 4's email was subject). User 4 has a significant increase in FN rate, possibly due to the classifier becoming biased towards predicting non spam over the period of the evaluation. Similarly, users 1 and 2 receive significantly more spam email than legitimate email and display a slight increase in the FP rate. For these users the system is being updated constantly to try to cope with the spam concept changes and as a result loses some accuracy in the prediction of legitimate emails. Increasing the value of k in the k -NN classifier may be a way of controlling this bias.

A shortcoming evident from the preliminary evaluation of ECUE was that it did not perform as well for users who receive high numbers of spam emails. These users had less than 90% of their email classified correctly. Users 1 and 2 in the full evaluation have this profile with the proportion of spam received varying from 76% to 89% respectively. Both of these users had over 93% of their mail classified correctly. (93.9% for user 1 and 95.3% for user 2). In addition the average error across both evaluations dropped from 8.15% in the preliminary evaluation to 6.8% in the full evaluation indicating that the additional level of learning, the periodic feature reselection process, allows the system to better handle the concept drift in the spam emails.

5 COMMERCIAL PROSPECTS

Our preliminary commercial analysis suggests that the commercial prospects for message classification systems that can learn are considerable. While this area of spam filtering is a rather crowded market there is a range of other areas where large volumes of messages need to be filtered or routed. There is considerable potential for message routing systems for incoming email into volume accounts such as info@company.com. Compliance legislation such as the Sarbanes-Oxley Act in the US⁷ generates a requirement to be able to monitor outgoing message streams to ensure compliance. There is also a considerable market for systems for routing XML messages in the financial sector. An attractive aspect of many of these application areas is that a low level of error can be tolerated. If a message arrives at the wrong destination it can simply be re-routed manually.

6 CONCLUSIONS

In this paper we describe ECUE, an example-based spam filtering application that learns from new examples of spam and legitimate email. We believe that this is a landmark deployment of an AI application that incorporates online-learning in an unobtrusive way. As a lazy local learner, ECUE offers distinct advantages over alternative eager approaches to spam filtering such as Naïve Bayes or Support Vector Machines, approaches that are more common in commercial filters. It provides capabilities to learn seamlessly without the need for a separate learning process. Also, the fact that spam is a diverse concept makes a local learner, an appropriate choice.

The evaluations show that ECUE is successful at filtering mail at a personal level, identifying between 92% and 94% of spam correctly with less than 1.0% false positives identified. For users who used ECUE as a second-level spam defense, operating the filter on email that had already been passed through a organisation-level gateway spam filter, ECUE still successfully filters between 48% and 66% of the spam correctly albeit with a slightly higher false positive rate for one of these users of just over 1%.

⁷ <http://www.sarbanes-oxley.com/>

To conclude, there is no single approach that will be 100% effective at handling spam. The solution to spam is currently a multi-layered approach, utilising legislative measures, authentication techniques and filtering. Filtering plays and will continue to play a significant role in this fight against spam. We believe that ECUE, our example-based approach to filtering can be an important contributor to content-based spam filtering. We have shown how it can handle the changes in spam emails with relative ease without putting too much burden on the individual using it.

REFERENCES

- [1] K Albrecht, N Burri, and R Wattenhofer, 'Spamato - an extendable spam filter system', in *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS'05)*, (2005).
- [2] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos, 'Learning to filter spam email: A comparison of a naive bayesian and a memory based approach', in *Proc of Workshop on Machine Learning and Textual Information Access, PKDD 2000*, pp. 1–13, (2000).
- [3] I. Androutsopoulos, G. Paliouras, and E. Michelakis, 'Learning to filter unsolicited commercial email', *Technical Report 2004/02, NCSR "Demokritos"*, (2000).
- [4] Z. Chuan, L. Xianliang, H. Mengshu, and Z. Xu, 'An lvq-based neural network anti-spam email approach', *SIGOPS Operating Systems Review*, **39**(1), 34–39, (2005).
- [5] S. J. Delany and P. Cunningham, 'An analysis of case-based editing in a spam filtering system', in *7th European Conference on Case-Based Reasoning (ECCBR 2004)*, eds., P. Funk and P. González-Calero, volume 3155 of *LNAI*, pp. 128–141. Springer, (2004).
- [6] S. J. Delany, P. Cunningham, and D. Doyle, 'Generating estimates of classification confidence for a case-based spam filter', in *Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005)*, volume 3620 of *LNAI*, pp. 170–190. Springer, (2005).
- [7] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, 'A case-based technique for tracking concept drift in spam filtering', *Knowledge-Based Systems*, **18**(4–5), 187–195, (2005).
- [8] H. Drucker, D. Wu, and V. Vapnik, 'Support vector machines for spam categorisation', *IEEE Transactions on Neural Networks*, **10**(5), 1048–1055, (1999).
- [9] K. R. Gee, 'Using latent semantic indexing to filter spam', in *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pp. 460–464. ACM Press, (2003).
- [10] A. Gray and M. Haahr, 'Personalised, collaborative spam filtering', in *Proceedings of 1st Conference on Email and Anti-Spam*, (2004).
- [11] J. Gómez Hidalgo, M. López, and E. Sanz, 'Combining text and heuristics for case-sensitive spam filtering', in *Proc of the 4th Computational Natural Language Learning Workshop (CONLL-2000)*, (2000).
- [12] L. Hughes, *Internet e-mail: protocols, standards and implementation*, Artech House Inc., 1998.
- [13] A. Kolcz and J. Alspector, 'Svm-based filtering of email spam with content-specific misclassification costs', in *TextDM'2001 (IEEE ICDM-2001 Workshop on Text Mining)*, pp. 123–130. IEEE, (2001).
- [14] M. Lenz, E. Auriol, and M. Manago, 'Diagnosis and decision support', in *Case-Based Reasoning Technology, From Foundations to Applications*, eds., M. Lenz, B. Bartsch-Sporl, HD. Burkhard, and S. Wess, LNCS, pp. 51–90. Springer-Verlag, (1998).
- [15] E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos, 'Filtron: A learning-based anti-spam filter', in *1st Conference on Email and Anti-Spam (CEAS 2004)*, (2004).
- [16] J. R. Quinlan, *C4.5 Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Mateo, CA., 1997.
- [17] J. Rennie, 'ifile: an application of machine learning to e-mail filtering', in *Proc of the KDD-2000 Workshop on Text Mining 6th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining*, (2000).
- [18] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos, 'A memory-based approach to anti-spam filtering for mailing lists', *Information Retrieval*, **6**(1), 49–73, (2003).
- [19] L. Zhang, J. Zhu, and T. Yao, 'An evaluation of statistical spam filtering techniques', *ACM Transactions on Asian Language Information Processing (TALIP)*, **3**(4), 243–269, (2004).

Knowledge-Based Recommendation: Technologies and Experiences from Projects

Alexander Felfernig¹ and Klaus Isak and Christian Russ²

Abstract. Recommender applications are used to support intuitive and personalized product and service selection processes for customers and sales representatives. In this paper we present a knowledge-based recommender environment which assists users in the identification of appropriate solutions from complex product and service assortments. We show how intelligent query relaxation, personalization and intuitive knowledge acquisition support the implementation of customer-friendly sales dialogues and report experiences gained in industrial recommender projects.

1 Introduction

Selecting from a large assortment of potentially complex items (e.g., financial services, computers, news, etc.) is still a challenging task since in many cases those items are offered using simple query interfaces under the assumption that customers have sufficient product domain knowledge [1]. Recommender technologies [2, 5, 11, 12, 16] improve this situation by providing support in the product selection process. An overview of different applications of recommender technologies can be found, e.g., in [13]. Compared to collaborative filtering [11] and content-based filtering approaches [3], knowledge-based recommender applications (advisors) [2, 5, 12, 16] exploit *deep knowledge* about the product domain for determining solutions fitting to the wishes of the customer. Using knowledge-based recommender technologies, the relationship between customer requirements and products can be explicitly modelled in an underlying knowledge base [5].

CWAdvisor³ is an environment supporting the effective development of knowledge-based recommender applications. The major innovations of CWAdvisor compared to other knowledge-based recommender approaches [2, 12, 16] are outlined below. Firstly, CWAdvisor includes a *graphical development and test environment* which supports the design, test and debugging of recommender knowledge bases and recommender process definitions [6, 5, 9]. This allows rapid prototyping processes by automatically translating graphical models into a corresponding advisor. Secondly, *model-based diagnosis techniques* [15, 6, 9] actively support customers in situations where no solution could be found. CWAdvisor can

identify *minimal relaxations* of filter constraints and determine a *minimal set of repair actions* for customer requirements, which guarantees that a solution can be found. Finally, recommender dialogues in CWAdvisor are *personalized* on the basis of an underlying finite state model [9] which describes possible interaction sequences. Using such representations, formulations of questions, answers, explanations, etc. can be personalized to the domain knowledge level and preferences of customers. Additionally, results of a recommendation process can be ranked using *multi-attribute object rating* [1].

We have successfully deployed a number of commercial recommender applications in domains such as financial services, e-government services, and typical e-Commerce application domains such as digital cameras, computers or TV sets. In order to show the utility of advisor technologies, we present an evaluation of deployed financial service recommender applications. Financial services recommendation is a knowledge-intensive task which in many cases overwhelms customers as well as sales representatives. Since the product assortment is quite manifold (investment, financing, pension, etc. - on an average about 40-60 products), many sales representatives focus on selling only a restricted set of products which leads to sub-optimal offers. The goal here is to identify a solution which fits to the wishes, needs and financial restrictions of the customer but also conforms to the sales strategy of the company. Furthermore, due to regulations of the European Union [17], financial service providers are forced to improve the documentation of customer advisory sessions. Detailed reporting is required which includes explanations as to why certain products were offered. For these reasons, financial service providers are extremely interested in tools providing an intuitive access to their product assortment.

The remainder of this paper is organized as follows. In Section 2 we introduce the CWAdvisor development and test environment. In Section 3 we discuss the set of AI technologies which are implemented in CWAdvisor. Finally, in Section 4 we summarize experiences from industrial projects and present an evaluation of successfully deployed applications.

2 System Architecture

The overall CWAdvisor architecture is depicted in Figure 1. Firstly, *CWAdvisor Designer* supports the graphical design of recommender knowledge bases. *Process Designer* allows the graphical design of recommender processes which describe the intended behavior of a recommender user interface [9]. *Test Designer* is responsible for the automated generation of test cases for recommender knowledge bases [8] and the interac-

¹ Institute for Business Informatics and Application Systems, University Klagenfurt, A-9020 Klagenfurt, Austria

² Configworks, Lakeside B01, A-9020 Klagenfurt, Austria

³ ConfigWorks Advisor - see www.configworks.com. Parts of CWAdvisor have been implemented within the scope of the project Koba4MS (Knowledge-based Advisors for Marketing and Sales) funded by the Austrian Research Fund (FFG-808479).

tive debugging of recommender knowledge bases [6] and recommender process definitions [9]. Finally, *CWAdvisor Server* is responsible for executing recommender applications.

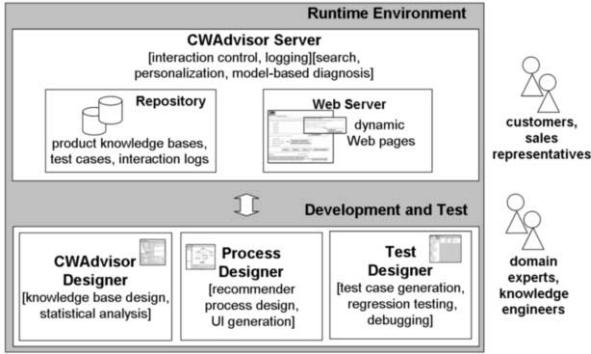


Figure 1. CWAdvisor Architecture.

CWAdvisor Designer. CWAdvisor Designer supports the graphical design of recommender knowledge bases. Such a knowledge base consists of the following parts [5].

1. Product properties are descriptions of the provided products, e.g., life insurances can be characterised by the possible length of life insurance policies, premiums of life insurance policies, or links to additional product documentation. Digital cameras can be described by their resolution, name of the producer, weight of the camera, etc.
2. Customer properties describe possible instances of customer requirements, e.g., within an investment advisory process the question *under the assumption that your investment of 10.000 EUROS decreases in value, at which value would you sell your investment?* is related to the willingness to take risks. In the digital camera domain, examples for customer properties are preferred motives (e.g., landscapes, portraits, etc.) or the maximum price of a camera.
3. Constraints restrict the possible combinations of customer requirements and products, e.g., return rates above 9% require the willingness to take risks or high resolution cameras are not provided in a low-price segment. Selection rules for products are denoted as *filter constraints*.

Process Designer. Besides the description of the relevant set of products, customer requirements and constraints, a process definition is needed to complete the model of a recommender application. A recommender process represents possible navigation paths which define the way the system adapts its dialogue style to the knowledge level and interests of the customer. Such process definitions are based on a specific type of finite state automaton [9] which represents navigation paths of a recommender application.

Test Designer. The increasing complexity of recommender knowledge bases makes quality assurance a critical task [14]. Within CWAdvisor, process definitions are the starting basis for automatically generating test cases for recommender knowledge bases [8]. Input sequences (examples for customer requirements) are automatically generated for selected paths of a recommender process definition, i.e., test case generation in CWAdvisor follows a path-oriented approach. A test case is the composite of one input sequence and a set of corresponding results. Generated test cases can then be presented to domain experts who decide on their validity (result

validation). Test cases that are considered as correct by the domain expert are used for future regression testing [10], i.e., if changes are made to the current version of the knowledge base, validated test cases are used to check the consistency of the new version of the knowledge base. On a more technical level, test case generation is based on the derivation of conjunctive queries from a given recommender process definition. For this purpose, a set of paths of a recommender process is selected. For each path in the set, a conjunctive query is generated and executed. Our experiences from developing recommender applications show that domain experts agree to accept the efforts related to the inspection of test sets since the quality of the recommendations is of serious concern, e.g., recommendations calculated for customers in the financial services domain must be correct in every case. The disposable time domain experts have for testing is restricted, therefore, we provide mechanisms which additionally reduce the number of test cases (see [8]).

Having completed the design and test of the recommender knowledge base and the recommender process definition, a corresponding recommender application can be automatically generated. An example for the user interface of a financial services recommender application is depicted in Figure 2.

3 Supporting Technologies

CWAdvisor supports a relational query-based approach, where a conjunctive query is dynamically composed from a set of filter constraints. The initial set of products to be presented as a solution to the customer is determined by executing the corresponding query on the product catalog.

Relaxation of Filter Constraints. Filter constraints describe the relationship between customer requirements and products, i.e., filter constraints determine which products are selected for a specific customer. A filter constraint is *active* if the given customer requirements are consistent with the precondition of the filter, e.g., if a customer is interested in short investment periods and a filter constraint enforces the selection of products with an investment period of less than two years in the case that a customer prefers short investment periods, the filter is active. There are situations where no solution can be found for a given set of customer requirements, i.e., the set of active filter constraints induces an empty set of selected products. In such a situation we support the *relaxation of filter constraints*. We apply concepts of model-based diagnosis [15] in order to identify a minimal set of filter constraints which have to be ignored in order to be able to calculate a solution fitting to a given set of requirements. Selection criteria for filter constraints are priorities, i.e., filter constraints with the lowest priority are first considered as candidates for a relaxation. Priorities can be defined a-priori by domain experts, or set during an advisory session.

Repair of Customer Requirements. In the financial services domain, in many cases filter constraints cannot be ignored or relaxed due to, e.g., legal restrictions or quality aspects of the recommended set of services. Simply reporting a retrieval failure, i.e., that no product matches all of the requirements, without making any further suggestions how to recover from such a situation is not acceptable or at least highly undesirable. Therefore, if we are not allowed to change the recommendation rules (filter constraints), we have to iden-

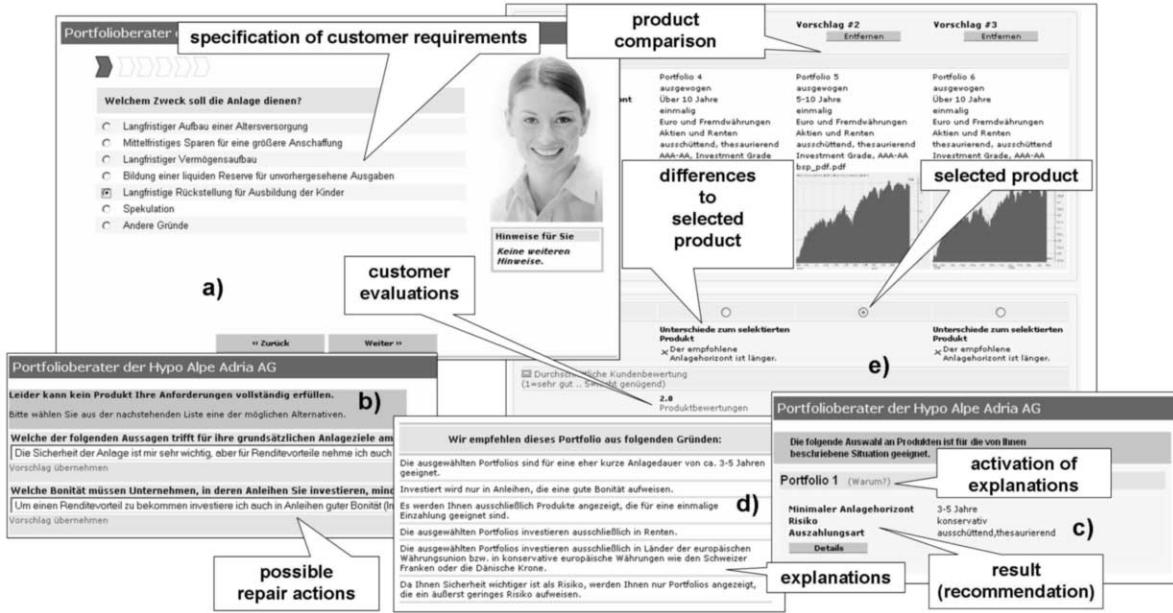


Figure 2. Example financial services advisor: questions are posed to customers and - depending on the given answers - further questions are determined (a). If no solutions can be found, a set of repair actions is presented (b). In the following, a set of alternative solutions including the corresponding explanations is presented (c,d). Finally, users can activate a product comparison component (e).

tify those requirements the customer is accepting to change in order to find a solution. Thus, our goal is to find a set of compromises (repairs) to be presented to the customer.

Similar to the relaxation of filter constraints, the computation of repair actions is based on concepts of model-based diagnosis [15]. The model-based diagnosis approach starts with the description of a system which is in our case the recommender knowledge base. If the actual behaviour of the system (set of customer requirements imposed to the knowledge base) conflicts with the intended system behaviour, the diagnosis task is to determine a *minimal* set of customer requirements which, when assumed to be retracted from the set of customer requirements, will explain the discrepancy between the actual and the intended system behaviour. Consequently, the goal is to identify a minimal set of variable settings in a given set of customer requirements which can be changed in order to identify a solution for the customer (i.e., the imposed customer requirements should be changed as little as possible).

An example for the calculation of repair actions in the financial services domain using model-based diagnosis is the following. Let $V = \{rr, wr, ip\}$ be the set of customer properties, where rr denotes the expected return rate ($\text{domain}(rr) = \{1-3\%, 4-6\%, 7-9\%, >9\%\}$), wr denotes the customers willingness to take risks ($\text{domain}(wr) = \{\text{low, medium, high}\}$) and ip denotes the investment period preferred by the customer ($\text{domain}(ip) = \{\text{short, medium, long}\}$) and $CR = \{c_1: \neg(rr > 9\% \wedge wr = \text{low}), c_2: \neg(rr > 9\% \wedge ip = \text{short})\}$ be the set of constraints in the recommender knowledge base. Furthermore, let $S = \{rr > 9\%, wr = \text{low}, ip = \text{short}\}$ be a given set of customer requirements ($S \cup CR$ is inconsistent). In this context, a repair is a *minimal* set of changes to S (resulting in S') s.t. $S' \cup CR$ is consistent. One possible repair S' for S is to change the requirement related to the return rate $rr > 9\%$ (to, e.g., $rr = 7-9\%$) which makes $S' \cup CR$ consistent. The second alternative is to change requirements related to the investment

period (e.g., $ip = \text{medium}$) and the willingness to take risks (e.g., $wr = \text{medium}$) which makes $S' \cup CR$ consistent as well.

Multi-Attribute Object Rating. A solution for a recommendation task is a set of products (e.g., financial services or digital cameras). The order of solutions should strictly correspond to the degree a solution contributes to the wishes of a customer (utility of a solution). CWAdvisor supports multi-attribute object rating [1], where each solution entry is evaluated w.r.t. a predefined set of dimensions. *Profit*, *availability* and *risk* are examples for such abstract dimensions in the financial services domain. Depending on the weighting of these dimensions for a specific customer (e.g., a customer is strongly interested in products with high return rates, i.e., compared to availability and risk, profit is important) solutions are ordered correspondingly (for details see, e.g., [5]).

Product comparisons. Comparison rules specify which argumentations are used to compare different products part of a recommendation result, e.g., if the risk rate of the selected (reference) product is lower than the risk rate of another product part of the result, then the comparison component should display the explanation *the risk level of this product is higher* (if product 1 is selected as reference product the explanation is given for all other products with a higher risk level).

Debugging Recommender Knowledge Bases. If a recommender knowledge base has been changed, regression testing (using Test Designer) is performed on the new version of the knowledge base. If there are (positive) test cases which are inconsistent with the new version of the knowledge base, a debugging process is triggered which calculates a minimal set of constraints to be changed in order to make the new version of the knowledge base consistent with the test base. The determination of such minimal sets is based on the application of model-based diagnosis (for details see [6]).

Debugging Process Definitions. In addition to the identification of faulty constraints in a recommender knowledge

base, we support the calculation of minimal sets of faulty transition conditions in recommender process definitions. When developing recommender user interfaces, mechanisms have to be provided which support the effective identification of violations of well-formedness properties of process definitions, e.g., if an input sequence reaches state q_i , there must be at least one extension of this input sequence to a final state (extensibility of input sequences). Path expressions form the basis for expressing such well-formedness properties on recommender process definitions. Details on the debugging support for developing recommender process definitions are provided in [9].

4 Experiences from Industrial Projects

Evaluations of Recommender Applications. We have deployed advisors in domains such as, e.g., financial services or digital cameras (see Table 1). In the following we present related evaluations.

Project	deployed in	number of users
Wüstenrot	2004	1.400
Fundamenta	2005	500
Hypo Alpe-Adria-Bank	2003	200 monthly
Geizhals	2003	10.000 monthly
Quelle	2005	600 monthly

Table 1. Examples for deployed recommender applications.

We have implemented financial services recommenders for online customers of the *Hypo Alpe-Adria* bank (www.hypoalpe-adria.at) (portfolio recommender and investment recommender). We evaluated those applications on the basis of an online questionnaire ($n=152$ participants, 23% denoted themselves as experts, 40% as beginners, the rest as having average knowledge in the financial services domain) where about 51% of the participants actively applied the recommender applications. Of those who applied the recommender applications, 98.5% were judging the provided recommender functionality as very useful when searching for financial service solutions. Within the scope of the questionnaire, the participants had the task to identify a specific type of product on the bank's homepage. In this context we have compared search times of those participants using a recommender application and those simply browsing Web pages of the bank (product fact sheets, etc.). In a comparison of the search efforts of the two groups we have identified a clear potential for reducing search costs. The average search time without recommender application support was about 10.5 min., a one-sample z-Test ($z=2.75$, $p < 0.01$) indicates that participants invested less time for product search if they used a corresponding recommender application. In addition to search efforts we also analyzed the subjective satisfaction with the provided advisory functionality of the homepage. The result of this analysis indicates an increased satisfaction with the overall advisory quality if a recommender is applied ($z=3.04$, $p < 0.01$). Additionally, CWAdvisor has been deployed for sales representatives of www.wuestenrot.at and www.fundamenta.hu. In both cases, financial service recommenders have been integrated into an existing Customer Relationship Management (CRM) environment that supports sales representatives in the dialogue with their customers (preparation, conduction and summary of sales dialogues). The motivation for the development of knowledge-based financial services recommender applications was twofold. Firstly, time efforts related to sales dialogues

should be reduced, i.e., time efforts related to the preparation and conduction of sales dialogues and time efforts related to the completion of sales dialogues (e.g., offer preparation). Secondly, a detailed documentation of advisory sessions should be supported in order to take into account regulations of the European Union [17] related to the documentation of advisory sessions in the financial services domain. On an average, a sales representative sells about 60-70 products per year, high performers, i.e., sales experts sell up to 500 products per year. One and a half years after the initial deployment of the recommender applications, we interviewed sales representatives of the *Wüstenrot* building and loan association ($n=52$), where about 23.1% of the interviewees were beginners (focus is on selling 1-3 different types of products), 28.9% were advanced sales representatives (selling more than 3 different types of products) and about 48.0% were experts (selling all products part of the assortment). Most of the sales representatives (81.8%) reported to use CWAdvisor functionalities throughout the sales dialogue or for the generation of documentations for completed advisory dialogues. Each such documentation includes a summary of customer preferences, a listing of recommended products, and an argumentation as to why the recommended products fit to the wishes and needs of the customer. In addition, about 53.8% of the sales representatives are applying financial service recommenders at home in order to prepare for the sales dialogue with the customer. In order to evaluate the satisfaction of sales representatives with the provided advisor functionalities, we have compared the evaluations of the CRM environment with and without advisory support. A one-sample z-test [4] resulted in a significant increase of the average level of satisfaction with the provided advisory functionality ($z = 10.83$, $p < 0.01$). No dependency exists between the expertise level of interviewees and the level of satisfaction with the provided advisory functionalities. On an average, interviewees specified the reduction of time efforts related to financial services advisory with 11.73 minutes per advisory session, where the majority of the reductions are explained by the automated generation of advisory protocols at the end of an advisory session and available summaries of previous dialogues at the beginning of a new advisory session. This corresponds to about 30.0% reduction of time efforts in the start phase and the final phase of an advisory session. Assuming that an average sales representative conducts about 170 advisory sessions per year, this results in a time effort reduction of about 33 hours per year for each person. A number of additional applications have been implemented on the basis of CWAdvisor, e.g., recommenders for www.quelle.at, one of the leading online selling environments in Austria and the digital camera advisor for www.geizhals.at, the largest price comparison platform in Austria.

Customer Acceptance of Advisor Technologies. Within the scope of COHAVE⁴ we have conducted a study [7] in which the key factors influencing the acceptance of knowledge-based recommender technologies by end-users were investigated. Examples for results of this first study within the scope of COHAVE are the following:

- Knowledge-based advisors clearly outperform simple product lists w.r.t. dimensions such as the overall satisfaction

⁴ COHAVE (Consumer Behavior & Decision Modeling for Recommender Systems) is a research project funded by the Austrian Research Fund (FFG-810996).

- with the advisory support, trust in that the advisor recommended the optimal solution or correspondence between recommendations and product expectations.
- Explanations of recommendation results significantly contribute to an increased trust in the recommended products, this is also associated with an increased perceived conformance between expected and recommended products.
 - Recommender versions including product comparison functionalities outperform all other versions (without comparison support) w.r.t. to all investigated variables. The reason for that is a lower mental workload of users when differences between products are clearly visually presented.

Knowledge Acquisition and Maintenance. Graphical knowledge acquisition is a major prerequisite for guaranteeing the maintainability and extensibility of recommender applications. In the financial services domain, the implementation and maintenance of knowledge bases must be supported for domain experts (typically non-programmers). Consequently, the knowledge acquisition component must provide intuitive and understandable modelling concepts. The most important functionality which has to be provided, is the automated generation of recommender applications. Domain experts want to experiment with different solution alternatives. This requires the support of prototyping-oriented development processes for recommender applications. In the case of www.wuestenrot.at, three domain experts are responsible for the development and maintenance of knowledge bases (investment & financing, pension & life insurance, property insurances).

We conducted a study with the goal to show the reductions of development and maintenance efforts triggered by the application of CWAdvisor automated debugging technologies. The study evaluates CWAdvisor technologies w.r.t. their advantages when identifying and correcting faulty constraints in recommender knowledge bases. Ninety-seven participants were confronted with a scenario in which they had to identify solutions for different types of error detection tasks. All the participants had experiences in the development of knowledge-based systems, especially experiences in implementing knowledge-based recommender applications. For each faulty knowledge base, one group of participants had to solve a diagnosis and repair task and one group (the control group) had only to solve a repair task with a predetermined diagnosis. Following this approach, we were able to compare efforts related to knowledge base development and maintenance with and without the support of automated debugging techniques. The major result of the study was that efforts related to the development and maintenance of constraints in recommender knowledge bases can, on an average, be reduced by about >30% when using an automated debugging support.

5 Conclusions and Future Work

Effective knowledge base development and maintenance and interaction mechanisms actively supporting users in the product selection process are crucial for successful recommender applications. In this paper we have presented the CWAdvisor environment which provides innovative development, maintenance and testing concepts allowing the effective handling of recommender knowledge bases. Furthermore, we have sketched the integration of model-based diagnosis and multi-attribute object rating concepts into the dialogue manage-

ment of recommender applications. These concepts allow the determination of minimal changes to a given set of customer requirements which can guarantee the retrieval of a solution and the ranking of recommendations in situations where more than one solution exists. The applicability of the presented concepts has been shown by an evaluation of successfully deployed advisors. There are different directions for future work such as, e.g., *automated test data generation and management* where we are working on the development of additional mechanisms supporting the selection of relevant test cases from a potentially very large set of candidates or the *integration of psychological theories from the area of consumer buying behavior* into design processes with the goal to increase acceptance and conversion rates of recommender applications.

References

- [1] L. Ardissono, A. Felfernig, G. Friedrich, D. Jannach, G. Petrone, R. Schäfer, and M. Zanker, 'A Framework for the development of personalized, distributed web-based configuration systems', *AI Magazine*, **24**(3), 93–108, (2003).
- [2] R. Burke, 'Knowledge-based Recommender Systems', *Encyclopedia of Library and Information Systems*, **69**(32), (2000).
- [3] R. Burke, 'Hybrid Recommender Systems: Survey and Experiments', *User Modeling and User-Adapted Interaction*, **12**(4), 331–370, (2002).
- [4] P. Dalgaard, *Introductory Statistics with R*, Springer, 2004.
- [5] A. Felfernig, 'Koba4MS: Selling Complex Products and Services Using Knowledge-Based Recommender Technologies', in *7th IEEE International Conference on E-Commerce Technology (CEC'05)*, eds., G. Müller and K. Lin, pp. 92–100, Munich, Germany, (2005).
- [6] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, 'Consistency-based Diagnosis of Configuration Knowledge Bases', *Artificial Intelligence*, **2**(152), 213–234, (2004).
- [7] A. Felfernig and B. Gula, 'An Empirical Study on Consumer Behavior in the Interaction with Knowledge-based Recommender Applications (to appear)', in *IEEE Conference on E-Commerce Technology (CEC'06)*, (2006).
- [8] A. Felfernig, K. Isak, and T. Kruggel, 'Testing Knowledge-Based Recommender Applications', *OEGAI Journal*, (4), 12–18, (2005).
- [9] A. Felfernig and K. Shchekotykhin, 'Debugging User Interface Descriptions of Knowledge-based Recommender Applications', in *ACM International Conference on Intelligent User Interfaces (IUI'06)*, eds., C. Paris and C. Sidner, pp. 234–241, Sydney, Australia, (2006).
- [10] G. Fleischanderl, 'Suggestions from the software engineering practice for applying consistency-based diagnosis to configuration knowledge bases', in *13th International Workshop on Principles of Diagnosis (DX-02)*, (2002).
- [11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl, 'Evaluating Collaborative Filtering Recommender Systems', *ACM Trans. on Information Systems*, **22**(1), 5–53, (2004).
- [12] B. Jiang, W. Wang, and I. Benbasat, 'Multimedia-Based Interactive Advising Technology for Online Consumer Decision Support', *Communications of the ACM*, **48**(9), 93–98, (2005).
- [13] M. Montaner, B. Lopez, and J. De la Rose, 'A Taxonomy of Recommender Agents on the Internet', *Artificial Intelligence Review*, **19**, 285–330, (2003).
- [14] A. Preece, S. Talbot, and L. Vignollet, 'Evaluation of Verification Tools for Knowledge-Based Systems', *International Journal of Human-Computer Studies*, **47**(3), 629–658, (1997).
- [15] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **23**(1), 57–95, (1987).
- [16] C. Thompson, M. Göker, and P. Langley, 'A Personalized System for Conversational Recommendations', *Journal of Artificial Intelligence Research*, **21**, 393–428, (2004).
- [17] European Union, *Richtlinie 2002/92/EG des Europäischen Parlaments und des Rates vom 9. Dezember 2002 über Versicherungsvermittlung*, Amtsblatt der EU, 2002.

Diagnosing Highly Configurable Products: Troubleshooting Support for Airbus Final Assembly Line

Andreas Junghanns and Mugur Tatar¹

Abstract. In this paper we describe the successful introduction of the model-based technology in a troubleshooting support tool for aircraft wiring under industrial conditions at Airbus, where strict cost/benefit analyzes decide over success or failure of a tool.

Aircraft built at Airbus Deutschland GmbH are highly customized products: virtually no two aircraft are identical. Troubleshooting support cannot be built in the traditional way of compiling static decision trees once and for all aircraft. Rather, diagnosis dialogs must be computed on demand for each individual aircraft.

Our solution is based on dynamically generating the diagnosis dialogs from on-line generated electrical signal nets. Generating diagnosis dialogs on demand is a considerable technical challenge. We have solved this with a combination of intelligent document extraction services and model-based diagnosis technologies. However, the paper focuses less on technical issues and more on the non-technological aspects that highlight the challenges of deploying well established intelligent technologies in industry.

For example, while existing engineering data is correctly interpretable by “fault tolerant and flexible” humans, it remains a challenging input for algorithms. Furthermore, some psychological aspects remain a challenge to the acceptance of the proposed tool.

1 Introduction

Providing troubleshooting support for highly configurable products is a challenge for the traditional methods based on “pre-compiled” troubleshooting manuals or decision trees. The model-based techniques [1] have the potential to overcome some of the problems in this context. However, specific challenges for this technology are the provision of cost-effective - if not fully automated - model building, and the reduction of the problem size to manageable dimensions. In this paper we describe the successful introduction of the model-based technology in a troubleshooting support tool for aircraft wiring under industrial conditions at Airbus. From a technological point of view, key solution features are: (i) the fully automatic model generation from on-line generated signal nets, and (ii) the problem size reduction based on the use of intel-

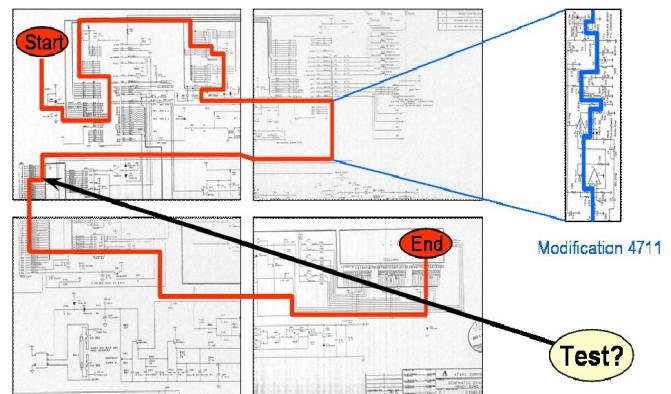


Figure 1. Example for Manual Wire Tracing

ligent documentation services that make use of connectivity information of components and subsystems and solve also all questions raised about the particular aircraft configuration.

2 Problem Environment

Airbus Deutschland GmbH assembles A318, A319 and A321 (short: single-isle) aircraft in Hamburg Finkenwerder from parts produced multi-nationally throughout Europe (Spain, UK, France and Germany) and delivers the completed Airbus to the client. The Final Assembly Line (FAL) is the central production unit for this process. Here is also where the testing of the aircraft’s electrical system is located. Even though all sub-assemblies are tested on their own, the aircraft is subjected to a rigorous, individualized, semi-automated system testing procedure. A computerized test system, called ESAO [4], is connected to the aircraft’s electrical system via multiple connectors and runs scripted test sequences. While executing a test script, ESAO uses peripheral devices

- to stimulate signals in the electrical aircraft system,
- to issue requests to the test personnel to manually change the state of the aircraft, e.g. to set switches or circuit breakers, and
- to read signals (sensors) from the aircraft’s system to compare against expected values.

Troubleshooting begins, when a difference to an expected value is detected. The following steps are currently undertaken

¹ DaimlerChrysler AG, Research & Technology, Berlin, Germany, E-mail: {Andreas.Junghanns,Mugur.Tatar}@DaimlerChrysler.com

by the test personnel:

- 1. Documentation:** Documentation is taken from a variety of sources, such as drawing databases, norm and standard documentation, applicable manufacturing rules etc.
- 2. Problem Reduction:** From all the available documentation the applicable (which aircraft) and desired (which subsystems) data needs to be extracted. Often, signals need to be traced by hand over several drawings of the specific aircraft currently under test (see Figure 1).
- 3. Diagnosis:** The test personnel can now change the state of the aircraft, stimulate and read signals in the electrical system in order to locate the faulty component(s).

Repair can begin after successful troubleshooting. No automated support or pre-fabricated information is currently available for the above process steps 2 and 3. Both troubleshooting steps are currently performed manually.

Airbus contracted DaimlerChrysler Research and EADS-M to analyze Airbus processes and suggest ways to increase the efficiency of the troubleshooting process at the FAL. Supporting the documentation phase, including the important wire tracing functionality, was the responsibility taken by EADS-M, while DaimlerChrysler Research assumed responsibility for improving the diagnosis process. A pilot (“proof-of-concept”) was implemented in 2003 to evaluate the suggested concepts and estimate the cost/benefit ratio for a production solution. The positive results lead to a contract to deploy a production version which was successfully installed in September 2005.

3 Project Analysis

At the beginning of the pilot project we concentrated our efforts on finding all the relevant *existing* data sources that were maintained well and could be expected to remain maintained in the foreseeable future. Our goal was to introduce troubleshooting support technology at the lowest possible cost, which means with existing data. Only when absolutely necessary we would consider gathering additional data. Then, after proofing the initial concepts, we could suggest possible extensions, including their cost/benefit estimates.

In a lengthy process of interviews and data analysis, we gathered information on data sources, internal data flows, information systems and responsibilities, compatibility of databases and future plans for these systems and their successors.

Furthermore, we spent several days at the final assembly line to learn about troubleshooting aircraft electrical systems first hand, getting to know tools and processes in order to understand how to support troubleshooting most effectively.

The data analysis interviews and visits at the FAL resulted in reports and recommendations and most importantly, in a list of solution requirements. The following subsections contain the most important of our findings.

3.1 Wire Tracing Data

Aircraft wire lists (AWL) and equipment lists are two fundamental ways to document in the aircraft industry. We were able to build on this information for the wire tracing functionality because most of the necessary information is contained in these lists.

Wire lists deliver connectivity information: Which wire connects which pins of which devices. In order to know which way the signal travels through a device, so-called shunt information is needed. This shunt information is part of the norm (standard) of the device. Equipment lists contain the part number of a specific device. Part numbers are usually norm identifiers which specify the device’s technical properties and behaviour, such as the time delay of relays, pin names and voltage requirements. Norm information is stored in so called master data.²

3.2 Norms and Specifications

While connectivity data is generally of high quality and therefore suitable for automatic model generation, we encountered challenges with the norm entries in the equipment lists, when reading it with our algorithms.

For instance, some low-volume, special order items do not have a proper norm. These devices are described by specifications to the supplier. Unfortunately, these specifications are usually drawings and documents. These drawing IDs are specified instead of a norm ID in the equipment lists. Furthermore, norm IDs are strings and are subject to different spellings, resulting historically from different national systems and styles.

Humans have no problem “solving” these irregularities because it is “obvious” to them how to distinguish between a norm ID and drawing ID, or how to adjust for missing/additional characters in a norm’s ID. In some sense, this data is correct given the contextual knowledge humans routinely have, but cannot be interpreted directly by a machine.

For the pilot project we added a few heuristic rules to catch a large number of the more obvious spelling styles. And still, we are unable to interpret 10-15% of the norm IDs. In these cases, the algorithms will detect a reference to an unknown norm and the wire tracing will stop at such a device prematurely, showing only part of the potentially larger signal net.

For the master data itself, pin information³ is generally available in machine readable form, shunt information is not. Because norms are not expected to change, we manually entered shunt information from standard sheets for the majority of the necessary norm IDs.

3.3 Diagnosis Data

For basic diagnosis functionality, nothing more than the wire tracing data is needed. However, with additional information, such as component fault probabilities, cost estimates for certain actions, such as state changes and tests, or signal types for each wire, the diagnosis dialog could be made more informed. However, and despite many attempts, we did not locate reliable, machine-readable sources for this kind of data and our solution had to do without these “nice-to-have” additions.

We analyzed close to 1000 troubleshooting reports and classified the problems found. We found out that if we could diag-

² For example, a norm would specify that pins A1 through A5 are all internally connected in a shunt of a terminal block. Or, that pins S1 and T1 of a relay are connected in neutral and S1 and U1 are connected when X and Y pins are subjected to a voltage drop.

³ Pin information specifies how many pins a device has and what their names are. In rare cases shunt information is encoded in those names.

nose “shorts to ground”, “broken components”, and “shorted to other signal”, we could support troubleshooting of well over 95% of all electrical problems. The remaining electrical “troubles” are failures such as wrong relay types (wrong time delay), resistor out of tolerance, temperature sensor defect or discharged battery.

In order to know where ESAO is connected to the electrical system of the aircraft, we used a database describing ESAO capabilities. Unfortunately, this system was designed for human use only. Therefore, substantial cleanup will be necessary in order to make this data complete and correct. We are currently using only a small subset of verified data.

4 Solution Description

This section sketches the technologies and tools used and the system architecture. The *documentation* and the *problem reduction* steps of the troubleshooting process are supported by the on-line intelligent documentation services. The *diagnosis* step is supported by on-line model-based diagnosis and test proposal services provided by our tool MDS [5].

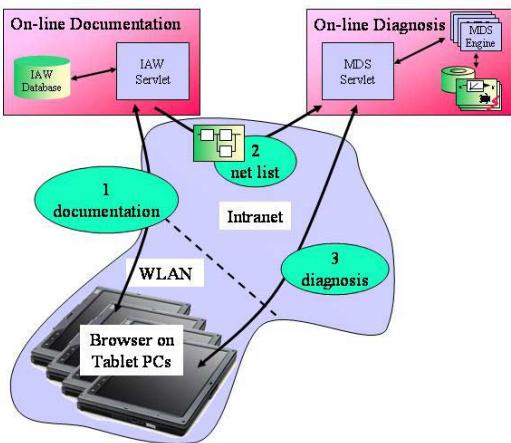


Figure 2. Solution Architecture

Figure 2 illustrates the solution architecture. The user interacts mostly with the remote documentation server (1). When requesting diagnosis support, the documentation system sends net lists to the diagnosis system (2). The user then interacts with the diagnosis support system (3).

4.1 Intelligent Documentation Services

The Customer Support Department at EADS-M in Ottobrunn has a long standing history in providing integrated product documentation technologies for different kinds of aircraft. Their key customer, the German Federal Armed Forces, deploys their Interactive Electronic Technical Documentation (IETD) software as stand-alone documentation without paper backup. The product documentation department imports data from engineering information systems into their production database. The import format is standardized by AECMA 2000 [3], providing an open interface to their software functionality. The production database integrates the data from

three nations and multiple engineering sources, verifies and re-compiles the data into internal database formats. This data contains all the necessary information about the individual aircraft, including configuration management and modifications. Error reports are generated when errors or conflicts are found.

The resulting database annotates the wiring data for fast retrieval. A highly configurable retrieval software offers multiple views on that data: classic wire and equipment lists, linked to master and norm data, aircraft manuals of all kinds.

The wire tracing functionality is important in our context. Given an arbitrary electrical component, all connected *signal nets* are generated and displayed. We call this component the *root of the signal net*. It is possible to focus the display on a single pin of the device. The resulting signal net is defined by all electrical wires and (simple) devices that carry the same signal. Nets are followed through wires, plugs and connectors, terminal blocks and splices, relays and switches to all meaningful signal ends. These are usually connectors into more complex devices. The display does not contain a classical wiring diagram, but a simplified network of devices and wires, annotated with device and wire IDs, offering pop-up menus with device properties, such as wire colour, gauge, EMC-class etc.

These nets can reach sizes of well over 400 wires connecting about 150 to 200 devices. However, these are manageable model sizes for the current capabilities of our model-based technology. Note that, this is a significant focus reduction compared to the tens of thousands of electrical devices and over 100.000 wires contained in each aircraft.

4.2 Model-Based Diagnosis Services

Model-based diagnosis is a well understood technology [1]. Its principles are beautifully simple and intuitive, and the technology's benefits are easily argued, theoretically.

In order to analyze a technical system using model-based technology, we need formally modelled component behaviour, e.g. based on qualitative or quantitative constraint models that capture physical and / or functional aspects of behaviours. Temporal aspects of behaviours are described by non-deterministic constraint automata in our system. Component type models are stored in a reusable model library. The system structure is needed in order to establish how components in a larger system interact with each other. Beside nominal behaviour, behavioural models for frequent faults, such as broken or shorted wires, are also specified. Assuming certain faults for components, we can evaluate the system model. All fault assumptions that produce simulation results consistent with the observed behaviour (measurements) are plausible faults. The diagnosis engine then proposes discriminating tests in order to reduce the number of alternative fault hypotheses. This troubleshooting cycle stops when either a unique fault is found, or no further tests reducing the set of plausible faults can be found. For an in-depth treatment of this technology see [2].

One of the advocated advantages of model-based technologies is the reusability of modelling effort when using model-libraries: Modelling the behaviour of an electrical component once allows its multiple reuse later, possibly parameterized to adjust the behaviour slightly. Usually, systems are assembled

by manually selecting and connecting components. When testing and diagnosing high-volume products, the cost for manual system assembly is easily amortized. However, at Airbus we are concerned with highly configurable and customizable systems - almost every aircraft is unique. The cost of manual system assembly would by far outweigh the benefits of diagnosis support. Therefore the solution used here is automated, on-line system assembly using components from a library and connectivity information from the on-line documentation services of the troubleshooting support.

When available, our system can deal with relatively detailed and complex component models that mix quantitative and qualitative constraints. However, there is a tradeoff between the additional benefit of using more elaborate models and the additional cost for acquiring (or cleaning up) the necessary data. Such an effort is in the present case not cost effective. The component models used here are rather simple, purely qualitative models - in effect, also the diagnostic support is quite basic. The bundle of the diagnosis technology with the product documentation doubles the benefit of the effort of maintaining the data and improves cost/benefit ratios.

The diagnosis and failure analysis software used in the project was developed over a period of several years and was already available before project start. A description of a previous version of the system can be found in [5], some specific aspects of the technology can be found in [6]. Numerous industrial projects and demonstrators for various clients have enabled us to create a powerful analysis tool fulfilling industrial requirements. Generation of guided diagnosis dialogs is but one functionality offered: others are decision tree generation, sensor placement, automated failure modes and effects (and criticality) analysis, model checking and others. This application only uses a fraction of the available functionality.

The test personnel, however, is never exposed to the complete tool functionality and complexity. A specially trimmed and simplified user interface was designed for the interaction with the test personnel (see Figure 3).

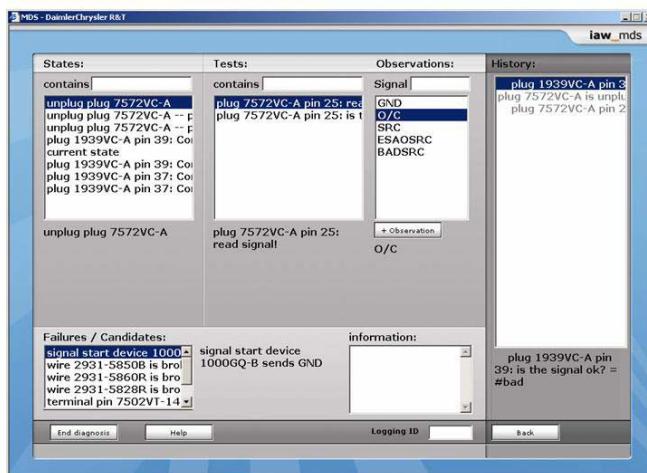


Figure 3. The Diagnosis Dialog

The user actions focus on the three central input lists:

1. Selecting one of the system states, possibly changing the

current state of the aircraft through control actions like changing the setting of a switch or by (un-)plugging a plug in left-most list,

2. Selecting a test from the second, sorted list, and
3. Selecting the result of the test in the third window.

Progress can be monitored in the lower left list of plausible failures. Adding more test results ideally reduces the number of plausible faults until no further discriminating tests can be found. Further features to support the user are history functions and filters.

This extremely simple, but highly intuitive dialog structure was deliberately chosen in order to make the basic functionality easily accessible to the tester and to avoid overwhelming them with too many complex features.

4.3 Uncertainty about the State of the World

Our diagnosis engine maintains consistent instantiations of the possible states of the system under test. Not only the correct or faulty states of the components are not known fully, also other state variables are incompletely known, for instance the connectedness of all switches, connectors, test equipment, etc. If one would inquire about all those details of the real-world state, the expected benefit of the system would quickly vanish, at least the users would run out of patience before the first viable suggestion would be made. Thus, assumptions have to be made. These additional state assumptions are treated by the diagnostic algorithms exactly like the correctness assumptions, but they are stripped out from the suggested diagnoses shown to the users. One way to deal with this uncertainty is to pick the most likely world state. Another one is to work with a superset of the most plausible worlds. We found the second approach to be the appropriate for user acceptance. Therefore we rather show too many test proposals and plausible faults than too few. Because the users know sometimes more about the state of the world, they can ignore inefficient or impossible choices.

4.4 Component Models

We were faced with a large number of Airbus components: tens of relays and switches and hundreds of typed terminal blocks and connectors. We have managed to reduce that set to a relatively small set of basic model elements. Let us consider the case of connectors as an example. We could have modelled all n-pin connectors explicitly. But this would not just mean an extremely large component library, but also many unused pins that are not part of the currently (reduced) signal net. Moreover, every pin of the connector, used or unused, increases the search space of possible faults by one more choice variable. Therefore, we compose connector instances from many basic pin models.

Detecting an unexpected symptom could also be caused by a problem in the test equipment, e.g. a broken wire from test computer to connector, a bent or broken pin of the connector. Since test equipment is under extreme stress because of its location in a production environment, such test equipment failures are more likely than aircraft electrical failures.⁴

⁴ The test personnel is first re-checking the setup of the test and its equipment, before starting a thorough troubleshooting session.

For all component types, beside nominal behaviour, we model mainly 3 types of component faults: shorts to ground, broken (no continuity), and connected to wrong signal (captures everything from badly isolated wire to bent connector pins). Again, we trust the intelligence of the tester; if we can blame a component with an abstract fault like “connected to wrong signal”, the tester has to be able to figure out the rest. The idea is to help narrow down the search to as few components as possible and leave the rest to the test personnel.

Because we generally lack quantitative information about signal types, we are using a finite set of generic signal types. If a wire or component is broken, that signal is interrupted (“O/C” is sent instead), and lack of continuity can be determined by comparing this value to the reference value, which assumes all components function properly.

Control actions, e.g. setting switches, unplugging and plugging connectors and test equipment, change the state of the system. They are part of the used component and system models based on the constraint automata framework used by MDS [6]. Test proposals, possibly also including required sequences of control actions, are generated for reading signals at all kinds of components, e.g. reading the signal on a pin or wire, or checking if an introduced continuity signal is present. Note, that reading the signal from a connector pin requires this connector to be unplugged (open). Since open connectors do not allow the signals to travel across, the maintenance of the correct state of the world, as affected by the (assumed or real) temporal sequence of control actions, is vital.

5 Technology Evaluation

The pilot implementation verified the technical feasibility of the solution. With a few changes, mainly to improve the flexibility and comfort of the user interactions, e.g. override system pre-sorting of tests, filters, undo and redo functions, we successfully built a production system. The feedback we received during training sessions with the test personnel strongly suggests that user acceptance will be high. Usage data is not yet available. Our tool helps to locate over 95% of all electrical faults. Systems exceeding 400 wires connecting around 150 to 200 devices can be diagnosed. Response times of the diagnosis engine is even for the largest systems within a few seconds.

Executing the test scripts with ESAO is a strictly guided and repetitive task. Many of the testers draw their job satisfaction from the intellectually challenging troubleshooting interludes. In fact, they pride themselves in their efficiency during troubleshooting. Providing a tool that reduces the challenge (fun) of the job does not find many friends.

Furthermore, efficiency is also a way to establish “seniority” among testers. Established hierarchies get challenged when helping the less efficient. Furthermore, the value of the tool for a senior tester is necessarily less than for a new employee. However, management is usually asking senior testers about their assessment of the tool.

It was interesting to see testers react to example diagnosis sessions with the pilot implementation. Humans are usually thinking linearly: When trying to find a discontinuity in a signal path, linear bi-section is the method of choice among most of the testers. This is also true because previously tracing a signal path was very tedious and very few testers bothered tracing the entire signal net (instead of the path alone).

But sometimes, it might be possible to exclude a number of plausible faults on the critical signal path by observing the signal off that critical path, for example because a connector out there is already open or more easily accessible. After observing the suggestions of the diagnosis engine a few times, the test personnel is usually quick to catch these patterns. And with the improved visualization given by the intelligent documentation system, this is also more intuitive. Of course, after the testers witness a couple of these patterns, it becomes increasingly harder to improve their efficiency with a guided diagnosis dialog. This training effect, especially for new test personnel, or radically new aircraft configurations, such as the A380, is an additional benefit for Airbus.

6 Summary

Building a real world application around a mature technology, such as model-based diagnosis, is still a non-trivial task. To achieve positive cost/benefit ratios, the use of existing data sources and the integration with the existing testing tools and processes is mandatory. Our solution takes this into account and demonstrates that even only with existing engineering data substantial troubleshooting support for highly varied products can be generated on the fly.

Packaging our solution with an intuitive, easy to use interface, hiding unnecessary details while maintaining usage flexibility, proved to have a vital effect on the user acceptance at Airbus.

7 Acknowledgements

The work presented here is an integration effort of two major software tools. Such an integration is impossible without the support of the teams involved. In Ottobrunn we would like to thank, first and foremost, Claudia Haubach-Lippmann and Johannes Gruber. In Berlin thanks go to the MDS team members Jakob Mauss and Volker May. We owe thanks to those supporting us at Airbus Hamburg: Olaf Kapitza, Martin Nagel, and Thomas Uhlendorf.

REFERENCES

- [1] J. de Kleer and B.C. Williams, ‘Diagnosing multiple faults’, *Artificial Intelligence*, **32**, 127–162, (1987).
- [2] *Readings in model-based diagnosis*, eds., Walter Hamscher, Luca Console, and Johan de Kleer, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
- [3] <http://www.aecma.org>.
- [4] <http://www.eads.com/web/lang/en/800/content/OF00000000400004/5/48/41237485.html>.
- [5] J. Mauss, V. May, and M. Tatar, ‘Towards model-based engineering: Failure analysis with MDS. ECAI-2000 workshop on knowledge-based systems for model-based engineering’, in <http://www.dbae.tuwien.ac.at/event/ecai2000-kbsmbe/papers/w31-05.pdf>, (2000).
- [6] M. Tatar, *Dependent Defects and Aspects of Efficiency in Model-Based Diagnosis*, Ph.D. dissertation, University of Hamburg, 1997.

Web-Based Tools for Codification with Medical Ontologies in Switzerland¹

Thorsten Kurz² and Kilian Stoffel³

Abstract. This paper presents three aspects through which code retrieval from an medical ontology can be improved. First, the recall of correct codes can be increased by the application of appropriate morphological operations on query terms and the enhancement of concept names with synonymous terms. Secondly, the efficiency of a manual selection process from a result set can be improved by structuring the result set correspondingly to the original ontology. Finally direct enhancements to the structure and content of the ontologies can be made in order to personalise it. These methods were implemented and evaluated in tools for two ontologies of major importance in the Swiss health care system: the ICD 10 and the TARMED ontology.

1 INTRODUCTION

The use of ontologies has a long history in the biological and medical domain. Currently there are two medical ontologies that are playing a major role for all hospitals in Switzerland and that are used for exchanging information with external entities.

1.1 ICD 10 and TARMED

One of the oldest medical ontologies still in use today is the International Classification of Diseases (ICD) [5]. Its structure and its content have evolved and through the years and its domain of application has widened from a pure classification of death causes to a statistical tool for the surveillance of diseases. The roots of the first version of this ontology go back to the late 19th century, the current version is the 10th revision, the ICD 10. This is the version that is used by the Federal Office of Statistics in Switzerland to record the frequencies of diagnosed diseases, thus hospitals in Switzerland are obliged since 1998 to codify their patient records with codes from the ICD 10 ontology.

The ICD 10 contains 26400 concepts that are organised in hierarchical chapters. For each concept there exists one title and many synonymous names. Each concept has a unique code, by which his place in the hierarchical structure can be found clearly without ambiguity. These codes are used to refer to the concepts. In contrast to ICD 10 the TARMED ontology has been created recently starting from zero [8]. The purpose of TARMED is the exchange of billing information between hospitals and health insurances. Its use is mandatory in the Swiss health care system since January 2004. The TARMED

ontology contains about 4600 concepts to codify therapeutic and diagnostic medical acts. These concepts are structured into 38 chapters with sub chapters, some of these concepts can be also found in the 20 service blocks or in the 36 service groups where the concepts are grouped according to different criteria like medical processes or medical qualifications. TARMED concepts have titles, but they have no synonymous names in the versions that have been released so far.

TARMED concepts have unique codes too, although here the codes do not indicate the place of a concept in the hierarchy. Selecting precise codes from ICD 10 or TARMED is crucial for obtaining precise statistics or correct hospital bills. However in discussions with medical doctors, we found a rather low motivation to spend much effort on searching correct codes in the ICD 10 ontology. This is not too astounding, considering that the coding is often done by young assistant doctors without proper training in coding, and the coding is usually done at the end of a busy day with no real perceived need for correct coding. Very much in line with our impressions, a study of the coding quality in the two cantons Vaud and Valais found that the accuracy of the applied principal diagnostic codes is as low as 56.5% in the canton Valais and 65.3% in the canton Vaud [6]. It is to be expected that better results can be achieved by offering tools that make coding easier and more efficient.

The fact that TARMED is used for billing is leading now to an increased interest in correct coding. In contrast to the Federal Office of Statistics, insurance companies are applying a strict validation with the help of various software that ensures that codes are accumulable among each other as well as compatible with the age and the gender of a patient and last but not least with the qualification of the medical staff. For the hospitals coding therapeutic or diagnostic acts with TARMED means now to keep the balance between billing too little due to overseen applicable codes and getting bills refused due to not allowable codes.

1.2 Practical issues with ontologies

While the application domains of the ICD 10 and TARMED ontologies differ, the practical issues with those ontologies are quite similar:

1. Using an ontology, either for coding or for document classification and retrieval necessitates the ability to locate concepts and codes in the ontology. This is not as obvious as it sounds, especially not if a user is not familiar with the structure of an ontology.
2. There are concepts that are mutually exclusive or can only be combined with a limited number of other concepts. Other concepts can only be used in combination with some prerequisite concepts.
3. In different hospitals, there is often a local terminology in use that

¹ This work was supported by the Swiss National Research Foundation: grant number 2100-066898.01/1

² University of Neuchâtel, Switzerland, email: thorsten.kurz@unine.ch

³ University of Neuchâtel, Switzerland, email: kilian.stoffel@unine.ch

is not exactly corresponding to the terminologies used in the ontologies.

4. Some hospitals do already have code-based accounting systems in place that have to be mapped to the new TARMED ontology.

This paper is about methods that can make codifying more reliable and more efficient and about the tools, which implement such methods. Section 2 presents morphological operations on query terms, Section 3 discusses how to personalise ontologies, Section 4 describes the resulting tools, and Section 5 evaluates the performance of a current prototype.

2 FINDING CODES AND CONCEPTS

A strictly hierarchical access to the concepts through chapters and sub-chapters can only be efficient, if the chapter titles are sufficiently explicit about their content, and even in this case it is possible that a code belongs semantically to more than only one chapter. To find a concept in an ontology in this way is far from obvious, especially if the user is not familiar with this ontology. In health care services, where a large part of the coding is done not by coding experts, but by young assistant doctors and nurses, this is one aspect that leads to bad coding.

Therefore it is a promising idea to allow for an alternative access to the concepts, based on keywords and other methods from the information retrieval (IR) domain [9]. Each concept has a name consisting of at least one, but most times several words that can be searched for. Like in a search engine a query field is provided, which is used to search concepts by their keywords.

For clustering the results of a query the hierarchical structure of ontologies turns out to be very useful. Instead of displaying the resulting concepts as a flat list, the concepts of the result set are presented in the hierarchical structure that is obtained by using the original ontology, and keeping only those branches of the hierarchy that contain a class of the result set. This is especially effective for large result sets with more than 10 results [4].

2.1 Morphological operations with medical terms

A well-known morphological operation in IR systems is stemming, which is applied to words in order to reduce variant word forms to common roots. This operation improves a systems ability to match query and document vocabulary. Basically, it consists in removing the plural, verb or attribute endings of words, e.g., the words *hopeful*, *hopefulness*, *hopes*, and *hoping*, are reduced to their stem *hope*.

Medical terms differ from common language words in such as that they are often combinations of several Latin or Greek words. Each of these words carries meaning and contributes to the meaning of the term. Sometimes such word parts can stand alone as well as being part of a more complex word. That means that for one medical concept there can be many more or less similar synonymous expressions. For example the following lines are synonymous in ICD 10:

1. Kératoconjonctivite
2. Kérato-conjonctivite
3. Kéратite superficielle avec conjonctivite

Here it would not be enough to simply cut off the endings. To be able to do a successful matching of the expressions above, it is necessary to cut the complex words into their meaningful parts. Consequently a stemming algorithm has been developed that reduces a medical terms into the smallest possible meaningful parts and cuts

off meaningless pieces. Basically the algorithm works similar to the Porter algorithm [7] with the difference that its structure is more modular, which permits to adapt it progressively to the characteristics of the endings of medical expressions. Additionally there is a component recognition module that uses a list of known word components of medical terms [3].

2.2 Proposing alternative query terms

The current version of our search engine returns all concepts in which all words in the query are present (logical conjunction, AND). Being too specific in a query can therefore be inhibiting and prevent any concepts from appearing in the results, whereas a less specific query might eventually return a result set with concepts that satisfy the requirements of a user.

Trying out different modified versions of a query manually is annoying and time consuming. A better approach is to let the system propose less specific, potentially successful alternative queries with less keywords in case that a query is unsuccessful and returns zero results.

3 PERSONALISING ONTOLOGIES

The hierarchical structure and the vocabulary used in an ontology reflect necessarily the application domain for which it was conceived and the world view of its editors. This is not much of a problem, if the application domain stays the same over the time and all users share the same world view. But there are different points of view for an health insurance company, for the accounting department in an hospital and for a medical doctor or a nurse describing their therapeutic acts. And such differences exist not only between different types of organisations, but also between organisations of the same type, e.g., there can be differences in the vocabularies that are used in different hospitals.

There are hospitals, which have established an internal extension to the ICD 10 concepts that provide a finer distinction of diseases combined with their own vocabulary. Most of the Swiss hospitals are also already using their own creations or modifications of classification systems for therapeutic acts for internal accounting. And of course those classification systems are far from compatible with the logic imposed by the TARMED ontology.

With all these differences it is clear that there is a need for a common ontology to provide a common ground and a shared language, but there is also need to maintain a localised, even personalised access to the concepts in an ontology.

3.1 Adding synonyms

If codes are retrieved through an interface that supports keyword queries, the easiest way to provide a personalised vocabulary is to allow personal expressions and keywords for each concept to be used.

This can be accomplished by either changing the concept names to user-specific names or by attaching user-specific synonyms to each concept. The second solution was implemented in the here presented retrieval systems. The integration of a medical synonym dictionary was considered too, but this approach turned out to be too unspecific, resulting in a considerably decreased precision for the queries.

3.2 Adding access paths

In TARMED we find parallel to the hierarchical organisation of concepts in chapters also service groups and service blocks that allow

to access the same concepts based on different criteria like different branches and qualifications.

Swiss hospitals most times have already local classification systems for therapeutic acts in use and contrary to TARMED the doctors and nurses are familiar with these systems. So the most obvious step is trying to translate the existing concepts to new concepts in TARMED. Due to differences in the definition of the concepts e.g. TARMED is quite specific about the qualification required to carry out a medical act, duration of the act and inclusions or exclusions of other concepts - there cannot be an automatic direct one to one translation from the existing local concepts.

However it is possible to reuse the structure of existing local classification systems and to add to each local concept the potentially applicable TARMED concepts. Or otherwise to define a new hierarchy in order to structure the TARMED concepts according to departments, services or other personal criteria.

To support this, we have created an web-based ontology editor that allows to add new concepts and relations between concepts to an existing ontology, to assign to concepts different labels and to modify them.

4 ONTOLOGY TOOLS

Prototypes for ICD 10 and TARMED code retrieval systems were build with high performance knowledge bases as backed [2, 1]. The systems implement the methods presented in Section 2 and Section 3 to various degrees and are accessible through standard web browsers. The ICD 10 Explorer (c.f. Fig. 1) was our first prototype system and its query processing is not as advanced as that of the TARMED Explorer, i.e. the ICD 10 Explorer does a matching of the query with the ontology and returns the highest ranking results, even if they do not match a 100% of the given keywords.

Figure 1. The ICD 10 Explorer. Appearances of the query terms in the ontology are highlighted yellow. The concepts in the red box in the upper half of the screen are references to other chapters that contain also matching concepts, but are mutually exclusive with concepts from the current chapter.

The TARMED Explorer on the other hand returns only results that match 100% of the given keywords, but it proposes less specific queries that would lead to results, if no results are found for the original query. Neither the ICD 10 nor the TARMED Explorer allow online editing and enhancing of ontologies so far, this is the domain of another system, the TARMED Editor (c.f. Fig. 2), which offers only ontology editing so far, but no keyword search.

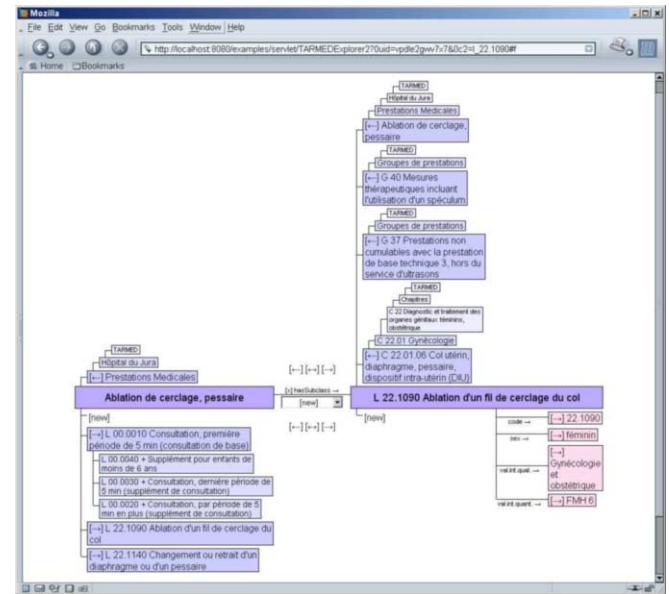


Figure 2. Mapping a local classification system for accounting to the TARMED ontology with the TARMED Editor.

5 EVALUATION

5.1 Code retrieval processes

The online prototypes of the ICD 10 and TARMED code retrieval systems have found many regular users in different hospitals in the French-speaking part of Switzerland and France. Among them is a French hospital that has made the ICD 10 system part of its coding procedure, so that we have currently the data of more than 40000 code retrieval processes to evaluate. An analysis of the log file shows a wide user base ranging from one-time users to users that come back frequently on a very regular basis (c.f. Table 1).

Table 1. Distribution of code retrieval processes per IP address (user).

Code Retrieval Processes per IP Address (User)	IP Addresses	Retrieval Processes
1-4	6811	(90.9%) 11211 (26.5%)
5-49	603	(8.1%) 8983 (21.2%)
50-499	68	(0.9%) 10526 (24.8%)
≥500	10	(0.1%) 11635 (27.5%)
total	7492	(100.0%) 42355 (100.0%)

A typical code retrieval process starts with a query from the user. The system uses the chapter hierarchy of an ontology to build a result tree and displays consequently only these chapters that contain codes that match the query. Then the user can start to browse through the result tree and eventually reformulate his query if necessary. All user actions are registered in a log file, which contains IP addresses, access times and dates, query expressions and the chapters and concepts that were visited. Thus we define a code retrieval process as a continuous sequence of queries and consequent browsing actions that are generated by one IP address, in which the queries have a similarity either in regard to their keywords or in regard to their results. *Continuous sequence* means that the sequence of actions is not interrupted for longer than a specific time interval. A visual log file

analyser gives a semi-structured view on those code retrieval processes (c.f. Fig. 3).

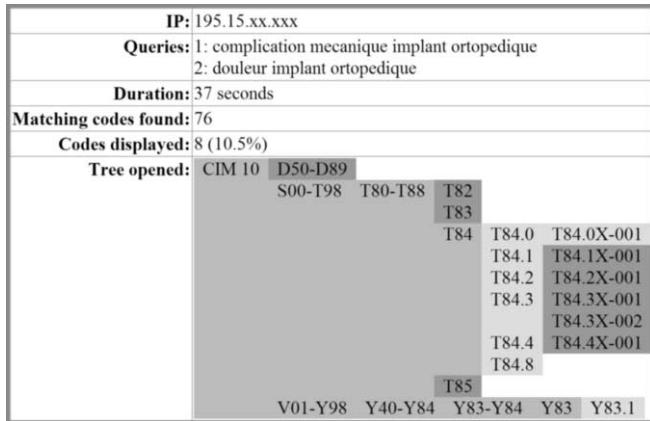


Figure 3. A semi-structured view of a code retrieval process with two queries that shows which parts of the result tree have been visited or ignored. Medium grey: Chapters of the result set that have been opened by the user. Dark grey: Chapters and codes of the result set that have been ignored by the user. Light grey: Codes that have been viewed by the user.

5.2 Efficiency, recall and precision

The average number of queries in a code retrieval process is 1.4, actually for 76% of the code retrieval processes only one single query was sufficient. The causes for code retrieval processes with larger query numbers are spelling errors, attempts to find a concept through a synonym that is unknown to the system and last but not least too unspecific queries that lead to too many results. Since the original report or bill that a user tries to codify remains unknown, it is never possible to make an absolute judgement if a returned code is correct purely based on the log data. Here it turns out to be helpful that results are not presented as a flat list, but as a tree structure. So instead of browsing through all results, the user decides which chapters seem interesting and worth opening. In fact usually the majority of chapters of the result tree is not visited, and this user behaviour can be used to distinguish between potentially correct and presumably incorrect results of query. An evaluation can be done based on the observation of which part of the result tree was visited, which one was ignored and which additional concepts were visited that were not part of the original result tree. Thus we define the *precision* of a result set as the percentage of visited concepts in comparison to the overall size of the result set. The precision is a measure for the relevance of a result set, the higher the value, the more relevant is the returned result set. Further we define the *recall* of a result set as the percentage of the visited concepts that were in the result set in comparison to the overall number of visited concepts during a code retrieval process. The recall is a measure for the completeness of a result set, the higher the value, the more complete is the returned result set. Recall and precision are here no absolute measures, but measures relative to a given system context, which allow to judge the impact of modifications of this system. Comparing those measures immediately after the first query and after the last query for all queries of code retrieval process together shows whether the additional queries really led to new information, i.e. increased recall and at what cost, i.e. decreased precision.

ICD 10 prototype uses a variant of the stemmer for French medical term which was discussed in Section 2. Based on the data from

Table 2. Performance of code retrieval systems with and without a stemmer module for medical terms.

	No Stemmer	Stemmer
Precision of first Query	40%	39%
Precision all Queries together	36%	34%
Recall of first Query	59%	79%
Recall all Queries together	65%	88%

the log file the code retrieval processes were reconstructed and sent to another system, which uses no stemming at all. Table 2 shows the comparison of the two systems. The usefulness of the stemming system becomes evident by the fact that the recall of the system with the stemmer reaches already at 79% after the first query, which is 14% better than a system without stemmer achieves with all the queries of a code retrieval process together. In each of the two systems can be observed that the further queries lead to increased recall (6%-9%), at the cost of some precision (4%-5%), but the recall of the system with the stemmer is 20% higher after the first query and 23% higher for the overall code retrieval process. The differences in terms of precision are far smaller, the system with the stemmer returns results that contain 1% to 2% less of relevant entries than the system without a stemmer. That means for accepting 2% more of not relevant codes in the result set, it contains 23% more of the codes that can be considered relevant for the request.

6 CONCLUSION AND FUTURE WORK

The development and testing of these tools is being carried out in close contact with medical professionals that are providing us with valuable feedback. Apart from positive feedback from those users, it was possible to quantify the benefits of a domain specific stemming algorithm. While most search engines are reluctant with the use of stemming because of fear of a considerable loss of precision, it could be demonstrated in case of the ICD 10 ontology that the loss of precision is smaller by factor ten compared to the gain of recall of potentially relevant codes. Considering the fact, that the user rarely knows the exact name of a code that he is looking for, there will be always several potentially correct answers to his query. Instead of forcing the user to browse through long lists of results, the hierarchical structure of ontologies allows to structure the results in a way that in average only 34% of the results are visited and taken into consideration by the user. This percentage becomes as small as 9% for large result set with more than 100 entries. The system doesn't try to be more intelligent than its users and therefore does not impose presumably "correct" answers on them, but presents information successful in a way that supports their decision making.

Both, the ICD 10 and the TARMED Explorer are currently running as stand-alone applications that can also be integrated in bigger frameworks as lookup-tools. The TARMED Editor will be evolving towards a multi-user editing environment and there is research going on to use the TARMED Explorer in combination with a voice recognition system for formulating the queries offline in a dictaphone.

ACKNOWLEDGEMENTS

We would like to thank Mr. Jean-Claude Vergriete for providing us with extensive lists of medical word components and synonyms and we also would like to thank the users of our system for their valuable feedback.

REFERENCES

- [1] C. Ciorascu, I. Ciorascu, and K. Stoffel, 'knOWLer - ontological support for information retrieval systems', in *Proc. of 26th Annual International ACM SIGIR Conference, Workshop on Semantic Web*, Toronto, (2003).
- [2] I. Ciorascu, C. Ciorascu, and K. Stoffel, 'Scalable ontology implementation based on knOWLer', in *Proc. of 2nd International Semantic Web Conference (ISWC2003), Workshop on Practical and Scalable Semantic Systems*, Florida, (2003).
- [3] T. Kurz and K. Stoffel, 'An advanced stemming algorithm for creating concept signatures of medical terms', in *Proc. of the ES 2001, Research and Development in Intelligent Systems XVIII*, pp. 273–281, Cambridge, (2001).
- [4] T. Kurz and K. Stoffel, 'Enriching ontologies for improved access to web documents', in *Proc. of IADIS WWW/Internet 2002*, pp. 815–818, Lisbon, (2002).
- [5] Organisation mondial de la santé, *Classification statistique Internationale des maladies et des problèmes de santé connexes*, OMS, Genève, 1995.
- [6] N. A. Najda, J. P. Vader, and A. Marazzi, 'Cohérence du codage diagnostique: étude de contrôle de qualité dans les hôpitaux vaudois et valaisans', *Médecine sociale et préventive*, (2306), (2000).
- [7] M. Porter, 'An algorithm for suffix stripping', *Program*, Vol. **14**(3), 130–137, (1980).
- [8] TARMED Suisse. <http://www.tarmedsuisse.ch>.
- [9] R. B. Yates and B. R. Neto, *Modern Information Retrieval*, Addison-Wesley-Longman, 1999.

Model-Based Failure Analysis with RODON

Karin Lunde¹ and Rüdiger Lunde¹ and Burkhard Münker²

Abstract. The model-based reasoning tool RODON supports engineers in their quest for reliable, well-designed technical products, by providing means to analyze system behavior, especially in case of failure, systematically. Based on a quantitative product model, it offers a wide range of analyses, including reliability analyses such as FMEA and FTA or the generation of diagnostic knowledge such as diagnostic decision trees. An object-oriented modeling language enhances the reusability of models and, together with an integrated design environment and extensive model libraries, considerably reduces the modeling effort. In this paper, we describe the modeling framework and the model analyses supported by RODON (considering as example the model of a comfort seat), and characterize the technology behind them. Finally, we discuss the experience gained during the development of RODON.

1 Introduction

During the life cycle of a technical product, various analysis tasks are necessary to ensure system safety and soundness of design, such as requirements analysis, design, reliability analysis, optimization and maintenance. Different teams specialized in certain phases of the development process have to work together in a highly distributed environment. The more people are involved, the more the exchange of knowledge becomes a bottleneck.

Model-based engineering aims to incorporate as much knowledge as possible in formalized, computer-interpretable models and to share those models between the different phases of the engineering process. The scope of those models is not fixed and can include very different aspects of structural and functional knowledge. To support knowledge reuse, domain specific, product specific, and task specific knowledge has to be separated. The goal of model-based engineering is to improve process efficiency as well as product quality by automation and to reduce errors caused by misinterpretations.

Since the introduction of CAD systems in the early 80s, the exchange of formalized structural product knowledge has continuously increased. Today, CAD models originating from the design phase are reused in almost all later phases, for example to export part specifications for subcontractors like wiring harness suppliers, or to support product management systems. There are also promising examples for the reuse of behavioral models, like model-based code generation for embedded control units (MATLAB tool suite) or failure and reliability analysis (MDS [9], AutoSteve[11], or AUTAS [10], to name only a few tools).

In this paper, we present the model-based reasoning tool RODON. It provides a wide range of analyses, with a focus on failure analysis

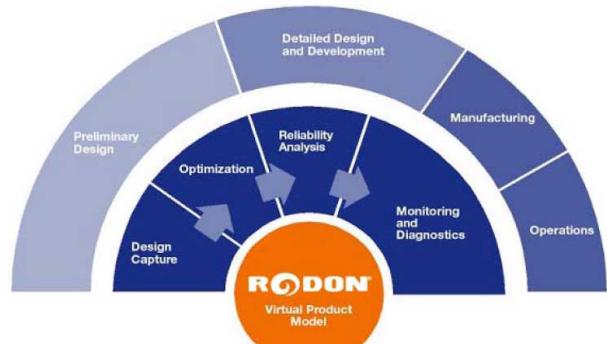


Figure 1. RODON's contribution to engineering tasks within the product life cycle

(see Figure 1). Based on a component-oriented, hierarchical model, which describes component behavior in terms of mathematical relations, RODON allows to

- simulate the behavior of the system in any state (which is part of the model);
- perform reliability analyses such as FMEA and FTA;
- compute optimal architectures for safety-critical applications;
- generate reliability block diagrams and failure trees;
- generate diagnostic knowledge such as diagnostic rules and diagnostic decision trees;
- perform interactive diagnoses with measurement proposal.

Since building and converting models between tools was identified as a major source of engineering effort, we attach great importance to the reusability of models. In the following, we describe the modeling framework and characterize shortly the algorithms which are at work behind the scenes to facilitate all those analyses. In Section 4, typical analyses are demonstrated considering as example a comfort seat model. A short discussion of lessons learned completes the paper.

2 Modeling in RODON

An object-oriented modeling language together with an integrated design environment and extensible hierarchical model libraries are the building blocks of a modeling environment which allows the engineer to create highly reusable and maintainable models with a moderate effort. In RODON, there are basically three modeling modes which differ widely in their degree of automation:

- The model topology can be imported automatically from CAD data, where component models are chosen from existing libraries.
- A model can be assembled graphically using existing model libraries (drag and drop).

¹ University of Applied Sciences Ulm, Germany, emails: k.lunde@hs-ulm.de, r.lunde@hs-ulm.de

² Sörman Information & Media AB, Heidenheim, Germany, email: burkhard.muenker@sorman.com

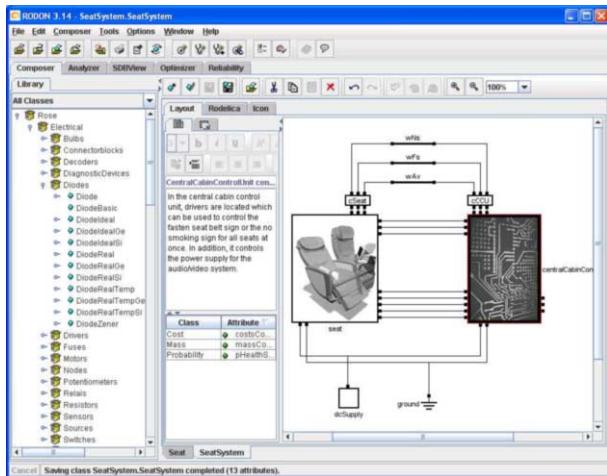


Figure 2. The model composer, at the left side the hierarchical library.

- A model can be built from scratch using the object-oriented modeling language Rodelica³.

In practice, these modeling approaches are frequently combined: The main structure of a model may be imported from a CAD tool, while some parts can be built re-using parameterizable library components, and company-specific parts have to be modeled by the engineer himself. The degree of possible automatization depends on how accurately the model structure matches the structure of the artifact. All models are organized within hierarchical libraries which can be extended continually by adding new models.

Object-oriented modeling Modeling in RODON is component-oriented, which means that the structure of the model reflects the structure of the artifact itself rather than the signal flow. In particular, this is essential for model-based diagnosis, where the engineer is interested in identifying the component whose failure caused the observed faulty system behavior. More important, a failure of one component can change the signal flow significantly, which restricts the use of a signal-flow-oriented model to certain behavior modes.

It is only a small step from a component-oriented to an object-oriented modeling paradigm. All essential properties of a component type are subsumed in a model class. It describes the interface of the component to other components, the substructure or the quantities which characterize its behavior, and their interactions in terms of constraints.

The concept of a model class is very general. Model classes represent components as well as connectors or physical quantities. They consist of attributes, behavior sections containing the constraints, and extends statements which declare base classes whose attributes and behavior are inherited. An attribute is defined by a name, which must be unique within the class, and a type, which is a model class (or an array of those). As primitives, Rodelica provides a set of built-in classes, e.g. Real or Interval. Besides, attributes and extends statements can contain parametrizations. Connections between the connectors of components are expressed by connect statements

³ Rodelica is a dialect of Modelica, which is a standardized object-oriented language to describe physical systems in a component-oriented and declarative way (see www.modelica.org). It differs mainly in the behavior representation, by using constraints rather than differential equations. This deviation from Modelica is due to the requirements of model-based diagnosis, where model behavior is often underdetermined (see [4]).

within the behavior section and translated into constraints at instantiation time.

The concept of inheritance greatly enhances the reusability and maintainability of model libraries: General knowledge can be encapsulated in more abstract base classes which are likely to be reused frequently. All changes which are made to a base class propagate automatically to all classes inheriting from it.

Uncertainty and constraints To solve diagnostic problems, explicit representation of uncertainty is needed. It may sometimes be hard even to predict a system's nominal behavior exactly. To scope all possible kinds of faulty behavior is in general infeasible. For some components, the number of all possible fault behaviors is infinite (consider e.g. a leak with arbitrary diameter in a pipe). And even if the number is finite, the number may be too high to be managed efficiently (e.g. shorted pins in a connector block).

Constraint techniques provide a convenient framework to represent this kind of behavioral knowledge. Arbitrary dependencies between values of different variables can be expressed by constraints which represent relations corresponding to the underlying physical laws. The representation is declarative. It abstracts from details how to perform simulations. The modeler does not need to direct relations by transforming equations into assignments or arrange sets of equations in a special order. It just suffices to formalize the relationships between the relevant variables. This abstraction makes it possible to arrange knowledge in a context-free component-oriented way. Especially the fact that constraint networks support reasoning about sets of possible values rather than exact values makes them well suited for diagnostic reasoning.

Constraint and data types The built-in classes can be understood as the data types provided by the modeling language. In RODON, the most important data types represent sets of integer numbers (built-in class Discrete), sets of boolean values (Boolean) or interval sets (Interval). Besides, the usual data types Integer, Real and String are available, as well as arrays and records. Several kinds of constraints are supported, providing the means to combine the quantitative and qualitative modeling approaches as appropriate:

- algebraic (in-)equalities,
- boolean relations (as formulas or truth tables),
- qualitative relations (tables),
- alternative behavior: conditional constraints and disjunctions,
- spline interpolation (for measured data or characteristics).

In some applications, it is adequate to use a directed modeling approach. For this reason, RODON supports assignments and algorithmic sections in models where values can be computed using elements known from programming languages, e.g. loops. Thus, the modeler can choose the modeling style and the level of abstraction which are particularly suitable for the task at hand with great flexibility.

An integrated design environment RODON's COMPOSER module supports the user in assembling, editing and debugging models. To the left, the library of the current project is shown. The user can choose between the compact tree view or a palette view, where the icons of selected library packages are listed. To the right, one or several class editors may be active (see Figure 2).

Each class editor features three main panels: the layout pane, the Rodelica pane, and the icon pane. In the layout pane, a model can be assembled graphically, by dragging classes from the library and

dropping them in the desired location on the canvas (thereby creating an attribute of the corresponding type), renaming and connecting them. A dialog assists the user in the parametrization, by showing the available parameters of the selected attribute and their admissible values. Besides, documentation texts can be assigned to the class as a whole and to each attribute. From the layout, a Rodelica text is generated, which can be viewed and edited after switching to the Rodelica pane. The Rodelica text and the layout view are synchronized — changes made in the Rodelica pane are visible after switching back to the layout pane. Finally, in the icon pane the user edits the external view at attributes which use the current class as their type, by assigning an icon and arranging connectors and labels around it. At any time, it can be checked whether the recent changes are syntactically correct and compatible with the project library.

3 Behind the scenes: The reasoning framework

The approach to model-based engineering followed by RODON is strongly inspired by the general diagnostic engine (GDE) [1] and its extensions for handling different behavioral modes [2] [13] and state transitions [14]. It is based on a reasoning framework which incorporates concepts from the classical approach to model-based diagnosis, such as component-oriented modeling, dependency tracking and conflict directed search for candidates. Reasoning is divided into two layers.

The first layer is the prediction layer. In this layer inferences are drawn to compute logical consequences based on a behavioral model and information about relevant system states (e.g. assumed fault mode assignments, symptoms, measurement results or top events). We describe technical systems as discrete-time systems. Constraints represent relations between different variables at the same point in time, and difference equations the evolution of state variable values from the current to the next point in time. During behavior prediction, two main analysis steps are performed in a loop.

Given value assignments of at least the dynamic state variables, the intra-state analysis tries to determine the values of all variables at a certain point in time. This step is an iterative domain reduction step. Based on the specified relations between the variables, the constraint solver successively prunes out those regions of the variable domains which do not occur in a solution.

The results of the intra-state analysis are fed into the inter-state analysis, which checks by evaluating the difference equations whether another state should be added to the current sequence, which value sets to assign to which state variable, and which time increment to use. By allowing both continuous and discrete time variables as state variables, events as well as continuous dynamic behavior can be simulated.

The second reasoning layer is the explanation layer. The methods applied here aim to find input parameter settings for the model such that it behaves consistently to some given external observations. In diagnosis, this includes at least assignment of a value to each fault mode variable. Additionally, a search in the possible space of other unknown inputs like switch states or hidden memory states can be included. Potential explanations are called *candidates*. The search for the most plausible candidates is determined by

- the definition of the candidate space, which is given by a set of variables and their possible values,
- the conflict information, which is obtained in the prediction layer by tracking the dependencies between the input parameters and their consequences during intra-state analysis, and additionally by

back-mapping [14] if a sequence of more than one state is analyzed, and

- some candidate orders defining minimality and plausibility.

Over the past years, this reasoning framework has been refined and extended in order to increase the reasoning efficiency without restricting the flexibility and expressiveness of the modeling paradigm. Most important are the following two modifications.

Inference completeness was improved by combining local consistency techniques with interval arithmetics, domain splitting and network decomposition (see [6]). The constraint solver extends the branch&prune algorithm (known from continuous constraint satisfaction problem solving, see [3]) by a network decomposition step and an interface to a reason maintenance system.

The reasoning efficiency was greatly improved by replacing the ATMS by a light-weight dependency tracking tool which we call the value manager (see [7]). In contrast to classical reason maintenance systems, the value manager actively controls resource consumption by applying data reduction techniques. Additionally, it incorporates strong focusing and data buffering.

A detailed account about the technology behind RODON can be found in [8].

4 Applications

4.1 Failure analysis of a comfort seat

In the following, we demonstrate several analyses provided by RODON by means of a simplified model of a comfort seat, like those in the first class of an aircraft. The seat model comprises an overhead unit featuring a reading light and visual signals, an audio/video unit, and three actuators to move the seat back horizontally and the foot rest both vertically and horizontally. Those elements are controlled by the operation unit which is situated in the arm rest. The seat control unit (SCU) contains drivers which process the incoming electric signals and drive the corresponding actuators or transmit messages via a bus to the central cabin control unit (CCU), as needed. The diagnostic trouble codes set by those drivers can be used by the service to locate faults.

In practice, the CCU controls all seats in the aircraft. From here, signals can be transmitted to all seats at once, e. g. to switch off the audio/video systems in case of a power drop. Each seat is connected to the CCU via a bus (modeled by electrical wires and connector blocks). The scope of the current model is restricted to only one seat and the CCU. Overall, the model consists of 183 components, 2540 variables and 2361 constraints.

Model-based diagnosis Consider a test of the comfort seat which involves checks of the three actuators, in the following order: Move the seat back forwards, then move the foot rest upwards, then move the foot rest inwards, and move the seat back backwards again.

When performing the test, it is observed that the seat back moves correctly, but the foot rest does not move at all. These observations, together with the switch positions in the four states of the test, are used as additional information for the diagnosis. Figure 3 shows the diagnosis: among all single faults there is only one candidate that can explain the system behavior in all four states, namely `seat.operationUnit.FootRestControl disconnected`; the next candidate is already a double fault. The dots at the end of the diagnosis denote the possibility of further double or higher order explanations. The diagnosis can be continued

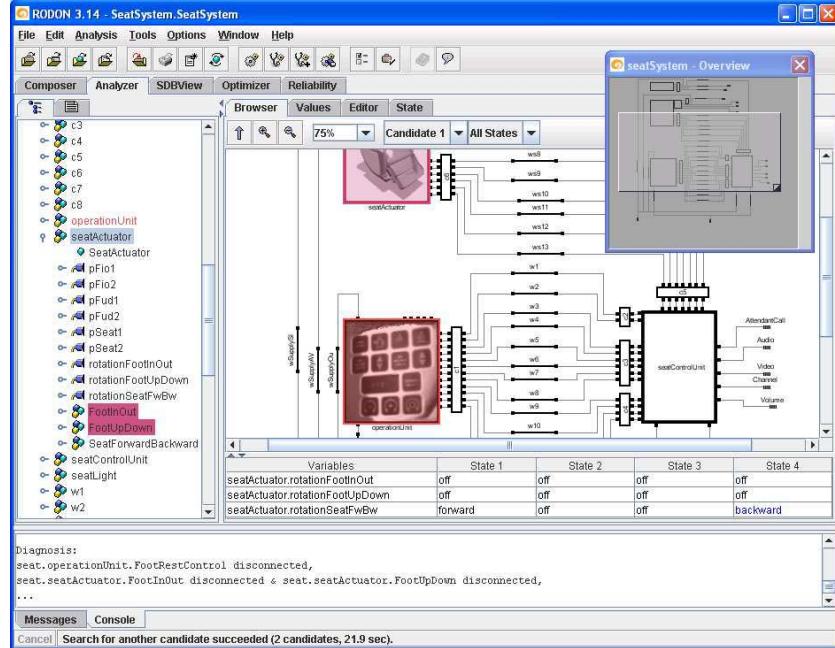


Figure 3. Model-based diagnosis of the seat system

upon request and would examine the candidates in the order of descending probability.

If the available information is insufficient to narrow down the possible explanations as far as in the last example, the tool is able to propose measurements whose results might help to eliminate candidates. Consider only the first two steps of the testing procedure above, with the same observation. Without the information about the outcome of the third test step, the model-based diagnosis finds alone 13 single faults which explain the malfunctioning foot rest. In such a situation, the interactive entropy-based measurement proposal may be used to restrict the set of candidates. The proposed measurements are listed in the order of their information gain, although the user is not bound to perform the measurements in that order.

Reliability analysis Comfort seats are not safety-critical systems, but frequent failures would annoy the passengers and are therefore to be avoided. A thorough reliability analysis in the design phase helps to reduce the probability of failures.

A common kind of reliability analysis is a failure mode and effects analysis (FMEA). RODON is able to generate the first columns of an FMEA table automatically. To this end, the user specifies which single faults and which operational states of the system are relevant for the analysis. The Cartesian product of all those operational states with the set of fault states (plus the state *System ok*) defines a so-called state space. An automatic simulation control module can then be used to perform simulations systematically for each state of the state space and to write the simulation results into a database, which we call state database (SDB). Finally, the desired table can be generated automatically, by evaluating and abstracting the SDB.

In Figure 4, a part of an automatically generated FMEA for the seat system is shown. The SDBVIEW module of RODON supports a great number of different views at the collected simulation results, among them a fault-oriented and an effects-oriented view. Customized table formats can be defined according to the requirements of the user.

4.2 Further applications

In addition to the systematic computation of an FMEA, the state database can be used to generate other knowledge representations, like diagnostic decision trees or diagnostic rules.

Diagnostic decision trees are used in the service bay to guide the mechanic through a failure analysis with minimal effort and costs. In a recent press release [12], Volkswagen confirmed that the RODON-generated test programs provide increased service quality concerning the maintenance of vehicle electronics.

Diagnostic rules represent a compact, compiled version of the model which can be evaluated efficiently. Therefore, diagnostic rules are well-suited for onboard diagnostics, where resources are usually scarce. For instance, the rules used in the onboard system's diagnostics (SD) of the Mercedes SL- and E-classes were generated by means of RODON.

For the reliability analysis of safety-critical systems, it is usually not enough to calculate the failure probabilities with respect to single faults (compare Figure 4). In such a case, RODON's special reliability module may be of use. For models which observe certain modeling conventions, it is able to compute minimal cut sets and failure probabilities for safety-critical functions algebraically. In addition, it generates the corresponding customary representations, namely reliability block diagrams and failure trees.

5 Conclusion

About seven years ago, RODON 3 started as a pure model-based diagnosis software but evolved gradually into a life-cycle support tool. The change in scope had mainly the following reasons, derived from our experience with various industrial customer projects:

- Besides the necessity to improve the diagnostic process with respect to flexibility and reliability, customer projects showed clearly that there is a growing need for automated diagnostic knowledge generation. Due to increasing system complexity, exploding variant diversity, and shortening product development

The screenshot shows the RODON 3.14 software interface with the title bar "RODON 3.14 - SeatSystem.SeatSystem". The menu bar includes File, Edit, Tools, Options, Window, Help, Composer, Analyzer, SDBView, Optimizer, and Reliability. The main window displays an FMEA table titled "FMEA (predicates) with rel. figu...". The table has columns for Fault, pHealthState, Effects, and Indications. The table lists several faults such as "cCCU disconnected", "cSeat disconnected", and various faults related to cabin control units and seat actuators. Each fault entry includes a probability value (e.g., 2.0001e-4t) and a list of effects and their corresponding indications.

Figure 4. Part of an FMEA of the comfort seat system.

To extend the FMEA table, the model has been enriched with probabilities, for all component fault modes in the model. During SDB generation, in every state a probability $pHealthState$ of the overall system is computed and written to the SDB, based upon the individual probabilities and with respect to the current fault state. As a consequence, this FMEA table provides, as an extra information (second column), a lower bound for the probability that a certain effect occurs.

cycles, diagnostic knowledge generation moves steadily from a practical experience-driven level towards a more theoretical level based on reasoning about construction data. For data processing and exchange, formal product description formats are needed, which cover structure as well as specified behavior. Component-oriented models like those used in model-based diagnosis are well suited for this purpose. To be able to serve as a vehicle for knowledge transfer, models must be generated in early stages of the product life cycle. But engineers which are involved in those early stages will spend time on modeling only if they benefit from this effort. This requires the modeling tool to provide additional functionality which helps them performing their own tasks.

- Modeling is always an investment since it consumes the expensive time of engineers. To maximize the return of invest, general reusable models should replace special purpose models in future, and be exploited wherever possible. Reliability analysis and diagnosis are related in many aspects. The needed product knowledge largely overlaps, and the level of abstraction in the knowledge representation is comparable. Additionally, common analysis algorithms can be used in both tasks. By extending the functionality of a model-based diagnostic tool to support reliability analysis, obvious synergy effects can be obtained.
- Compared to pure simulation approaches, our constraint-based approach with its ability to represent fuzziness and uncertainty explicitly gives the modeler more flexibility in his choice of abstraction level. This includes very high level abstractions, possibly qualitative ones, which are especially well-suited for very early stages of the product life cycle.

Our approach is well suited for realistic engineering applications (see also the examples in [5] and [8]). The main functional advantages in comparison to pure simulation approaches are the support of conflict-directed search strategies by dependency tracking, and the coverage of ranges of possible behaviors by set-valued reasoning about system behavior.

This additional functionality is not provided free of charge. For behavior prediction, pure simulation engines are significantly faster than the algorithms used within RODON. Although reason maintenance can compensate part of this difference, performance still remains a major issue when faced with the analysis of large systems.

Further improvements seem to be possible by combining numerical with algebraic methods. Especially in component-oriented models of electric circuits, a high amount of constraints consists of simple linear equations, which can easily be simplified by algebraic transformations. Therefore, the integration of such algebraic transformations into the reasoning framework seems to be a promising direction of further investigation.

REFERENCES

- [1] J. de Kleer and B. C. Williams, 'Diagnosing multiple faults', *Artificial Intelligence*, **32**, 97–130, (1987).
- [2] J. de Kleer and B. C. Williams, 'Diagnosis with behavioral modes', in *Proceedings of the IJCAI'89*, pp. 1324–1330, (1989).
- [3] P. Van Hentenryck, D. McAllester, and D. Kapur, 'Solving polynomial systems using a branch and prune approach', *SIAM Journal on Numerical Analysis*, **34**(2), 797–827, (April 1997).
- [4] K. Lunde, 'Object-oriented modeling in model-based diagnosis', in *Proceedings of the First Modelica Workshop, Lund*, (2000).
- [5] K. Lunde, 'Ensuring system safety is more efficient', *Aircraft Engineering and Aerospace Technology*, **75**(5), 477–484, (2003).
- [6] R. Lunde, 'Combining domain splitting with network decomposition for application in model-based engineering.', in *19th Workshop on (Constraint) Logic Programming W(C)LP 2005*, ed., A. Wolf et al., Ulmer Informatik-Berichte, (2005).
- [7] R. Lunde, 'Introducing data reduction techniques into reason maintenance', in *Proceedings of DX06*, Burgos, Spain, (2006).
- [8] R. Lunde, *Towards Model-Based Engineering – A Constraint-Based Approach*, Ph.D. dissertation, Universität Hamburg (to appear), 2006.
- [9] J. Mauss, V. May, and M. Tatar, 'Towards model-based engineering: Failure analysis with MDS', in *Proceedings of ECAI 2000, Berlin, Germany*, (2000).
- [10] C. Picardi, L. Console, and F. Berger et al., 'Autas: A tool for supporting fmeca generation in aeronautic systems.', in *Proceedings of ECAI 2004, Valencia, Spain*, pp. 750–754, (2004).
- [11] Chris Price, 'AutoSteve: Automated electrical design analysis', in *Proceedings of ECAI 2000, Berlin, Germany*, pp. 721–725, (2000).
- [12] Volkswagen Media Services. Reliable Volkswagen-electronics by new diagnostics - Improved service quality in VW workshops. press release, February 2006. <http://www.volkswagen-media-services.com>.
- [13] P. Struss and O. Dressler, 'Physical negation: Integrating fault models into the general diagnostic engine.', in *Proceedings of IJCAI 1989*, pp. 1318–1323, (1989).
- [14] M. Tatar, 'Diagnosis with cascading defects', in *Proceedings of ECAI 1996, Budapest, Hungary*, pp. 511–518, (1996).

This page intentionally left blank

9. Perception

This page intentionally left blank

A Learning Classifier Approach to Tomography

Kees Joost Batenburg¹

Abstract. Tomography is an important technique for noninvasive imaging: images of the interior of an object are computed from several scanned projections of the object, covering a range of angles. Traditionally, tomographic reconstruction algorithms have been based on techniques from analysis and linear algebra. In this paper we describe how a particular version of the tomographic reconstruction problem, known as *discrete tomography*, can be considered as a *classification problem*. By making this connection between two seemingly unrelated scientific areas, the full machinery of learning classifier theory can be used to solve tomographic problems.

The use of classifiers that can be trained from examples for tomography has two main advantages. First, prior knowledge concerning the images of interest can be learned automatically. Second, there are several types of classifiers that can perform the classification task very fast once trained. Real-time reconstruction is one of the main goals in tomographic imaging.

Tomography typically deals with huge amounts of high-dimensional data, which makes the corresponding classification task very hard. We describe several steps to reduce the dimensionality of the input data. For one type of classifier, the multilayer perceptron, we provide experimental evidence showing that it can indeed be used for efficient tomographic imaging.

1 INTRODUCTION

The field of tomography deals with the reconstruction of objects from their projections [4, 8]. In this paper we focus on *transmission tomography*, where projections are obtained by letting a beam (e.g., X-rays, electrons, neutrons) pass through the object of interest. Interaction of the beam with the object attenuates the beam, which is measured by a detector array after it has passed through the object. Figure 1 shows the basic idea of tomography. The attenuation of the beam along a certain line depends on the *materials* that the line passes through and the *lengths* of the corresponding line segments.

Since the start of its development in the 1970s, X-ray tomography has become a common tool in medical imaging for noninvasive scanning of patients. More recently, transmission tomography has been used extensively in electron microscopy for computing 3D reconstructions of samples [7]. Tomography is also used often in industry, for nondestructive testing of manufactured objects or reverse engineering.

Efficient algorithms are available for computing tomographic reconstructions. *Filtered backprojection* (FBP) is widely used, because of its simplicity and efficiency (see, e.g., Chapter 3 of [4]). The algorithm is based on the relation between the 2D Fourier transform of the unknown image and the 1D Fourier transforms of the measured

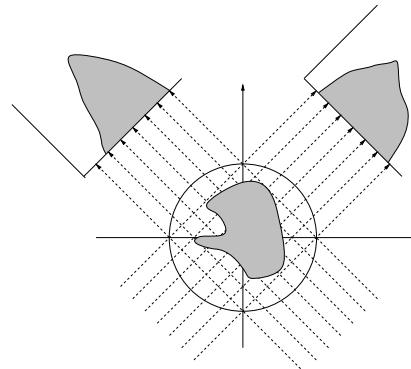


Figure 1. Basic idea of parallel beam transmission tomography.

projections. We refer to FBP and similar algorithms as *analytic reconstruction methods*, as the FBP algorithm is based on real analysis.

Another popular class of reconstruction algorithms uses methods from linear algebra to solve large, sparse systems of linear equations. If we represent the unknown image as an array of pixels then each measured line projection defines a linear equation on the values of these pixels, yielding a system of linear equations. Several iterative methods are available for solving this system, notably ART, SART and SIRT (cf. Chapter 7 of [4]). We refer to such methods as *algebraic reconstruction methods*.

Both analytic and algebraic reconstruction methods are *continuous* tomography algorithms, which means that the reconstructed image may exhibit a wide, continuous spectrum of gray values. Continuous reconstruction algorithms typically require a large number of projection angles to compute accurate reconstructions, i.e., more than 50. Reducing the number of projections is important, as it reduces the scanning time and – in the context of medical imaging – the radiation dose for the patient. One way to achieve this goal is to use additional application-specific prior knowledge in the reconstruction algorithm. It is desirable to automate the discovery of such prior knowledge, as the discovery process is usually difficult and time-consuming to do by hand.

The field of *discrete tomography* (DT) focuses on the reconstruction of images that consist of a small, discrete set of gray values that are known in advance [3]. Examples of such images can be found in semiconductor industry (semiconductor parts consist of few materials), medical imaging and crystallography. By assuming the discrete nature of the set of gray values it is often possible to vastly reduce the number of required projections. So far, most research in DT has been focused on the reconstruction of binary images (i.e., black-and-white). To reconstruct binary images from very few projections (i.e., less than 10), more prior knowledge has to be used, such as smoothness assumptions. At present, this knowledge is usually modeled by

¹ Leiden University and CWI, Amsterdam, The Netherlands, email: kbatenbu@math.leidenuniv.nl

hand.

The DT reconstruction problem can be considered as a classification task: there is a small number of classes (the gray values) and each pixel has to be assigned to one class, based on the projection data of the image. If a learning classifier can be trained for this task, using a set of application-specific training images, prior knowledge could be learned by the training algorithm instead of being modeled by hand. Another important benefit of using learning classifiers for DT is that the classification task can usually be performed extremely fast once the training phase is complete. This is particularly important for *real-time* imaging of dynamic processes [5]. Current DT algorithms typically take more than one minute to reconstruct a single slice of a 3D volume, which makes real-time imaging infeasible.

As an example, consider the problem of reconstructing a raw diamond slice from its projections (see Figure 2). A diamond consists of only one material, which leads to a binary classification task for the pixel values. The fact that the continuous reconstruction from Figure 2 is not completely binary, is the result of using too few projections and noise in the measured projection data.

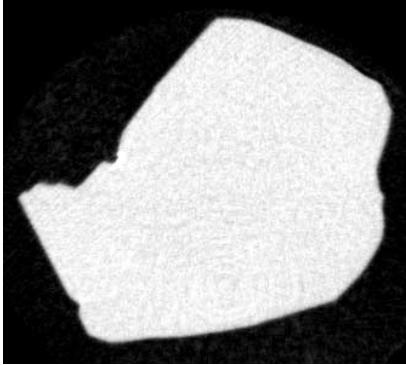


Figure 2. Slice of a raw diamond, reconstructed by continuous tomography from 125 projections; courtesy of DiamScan, Antwerp, Belgium.

Although the number of classes for the DT classification task is small, the dimension of the input space can be huge. Each point in the input space contains a set of measured line projections. If 10 projections are used, and each projection consists of 256 line projections, every input pattern for the classifier is a real-valued vector of 2560 elements. To make the problem even harder, the dependency of each pixel on the projection data is different, so it seems that a new classifier has to be learned for each pixel.

In the next sections we describe a series of steps that reduce the input data, making its size suitable for applying learning classifier approaches. We also show that by applying a suitable transformation to the input data, the same classifier can be used for all pixels, instead of having to learn a new one for each pixel. Not only does this transformation decrease the training time, but it also vastly decreases the number of training images that is required, as each pixel in a training image can now be used as a separate training example by the training algorithm. We present reconstruction results, using a multilayer perceptron as a classifier, which demonstrate that using a small set of training images already leads to good reconstruction results.

2 RECONSTRUCTION PROBLEM

Figure 3 shows the main setting of 2D parallel beam transmission tomography. The object of interest is contained within a circular

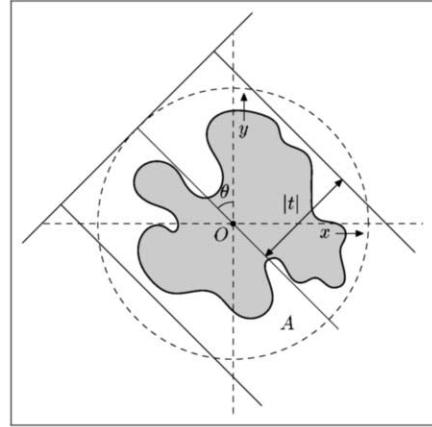


Figure 3. Schematic view of the experimental setup used in parallel beam tomography.

area A , the *imaging area*. A parallel source and detector array rotate around the object, measuring the projections from several angles. We assume that the object is *binary*, i.e., that there are only two gray values, for the interior and exterior of the object. In Section 5 we will discuss the generalization to more than two gray values.

The image of the unknown object that we want to reconstruct is considered as a mapping $f : A \rightarrow \{0, 1\}$. A value of 0 (black) denotes the exterior of the object and 1 (white) denotes the interior.

Projections are measured along lines $l_{\theta,t}$ of the form

$$l_{\theta,t} = \{(x, y) \in \mathbb{R}^2 : x \cos \theta + y \sin \theta = t\}.$$

The line $l_{\theta,t}$ makes an angle θ with the vertical axis and is at distance $|t|$ from the origin.

The *Radon transform* $P_\theta : \mathbb{R} \rightarrow \mathbb{R}$ for projection angle θ of the image f is defined as

$$P_{\theta,f}(t) = \int_{l_{\theta,t}} f(x, y) \, ds.$$

The function $P_{\theta,f}(t)$ measures the intersection length between $l_{\theta,t}$ and the interior of the object. It is also called the *projection* for angle θ and its values $P_{\theta,f}(t)$ are called *line projections*.

In practice, the function $P_f(\theta, \cdot)$ is usually not measured in single points t . Instead, the total projection in a strip covering a small t -interval (t_ℓ, t_r) , is measured as

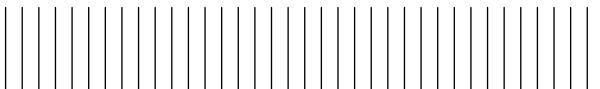
$$S_{\theta,f}(t_\ell, t_r) = \int_{t=t_\ell}^{t_r} P_f(\theta, t) \, dt.$$

The function $S_{\theta,f}(t_\ell, t_r)$ denotes the intersection area of the strip between l_{θ,t_ℓ} and l_{θ,t_r} with the unknown object. Typically, the value $S_{\theta,f}(t_\ell, t_r)$ is measured for consecutive strips of fixed width. In our approach we also need to evaluate $S_{\theta,f}(t_\ell, t_r)$ for other values of (t_ℓ, t_r) . These values are computed by linear interpolation of the measured projection data. The *tomographic reconstruction problem* deals with finding a function $f : A \rightarrow \{0, 1\}$ that adheres to a measured set of strip projections. Its solution is not necessarily unique. If the number of angles for which the projections are given is small there may be many, very different solutions. Therefore additional prior knowledge must be used in the reconstruction algorithm to obtain satisfactory results.

3 RECONSTRUCTING A SINGLE PIXEL

We now focus on the task of reconstructing a single pixel in a binary image from the given projection data. If the image has some local structure, such as smoothness, it may seem to be better to reconstruct all pixels simultaneously, as the gray value of a pixel depends on the gray values of its neighbours. However, the gray values of neighbouring pixels also depend on the projection data. Therefore, dependencies on neighbouring pixels can also be modeled as dependencies on the projection data.

In principle, it is possible to train a classifier by performing reinforcement learning directly on the measured projection data. Consider a typical discrete tomography problem, where 256 strip projections are measured from 20 angles. In that case the dimension of the input space for the classifier is over 5000, which makes the training task very difficult. Our approach to dealing with the high dimensionality is to perform a preprocessing step that maps the projection data to a lower dimensional space, while still capturing most of the important features. The intuitive reasoning behind this reduction is that the projection data for strips that go through a certain pixel are much more important to the value of this pixel than strips that are far away. To reconstruct a pixel, we use fine measurements near this pixel and coarse measurements further away. As an example, consider the case that we are only interested in reconstructing the central region of the imaging area. Figure 4a shows the strips as they are measured by the detector array. Since we only need a high quality reconstruction of the central region, we can use coarse strips outside this area, as in Figure 4b. By merging several consecutive detector values into a single ‘‘coarse’’ detector, the strip projections for the configuration in Figure 4b can be computed from the measured, fine strip projections. One may also wonder if we can leave the distant strip projections out completely, see Figure 4c. Figure 5 shows a binary phantom image (i.e., a test image) and three SIRT reconstructions from 30 projections, using the three strip configurations from Figure 4. On the right, a magnification of the central region is shown.



(a) Equally spaced fine strips.



(b) Fine strips covering the central region, exponentially increasing strip sizes outside



(c) Fine strips in the central region, no strips outside.

Figure 4. Three sets of detector strips that can be used to reconstruct the central region of the imaging area.

The reconstruction in Figure 5b was computed using all available strip projection data. The reconstructed image in Figure 5c, which was computed using very coarse projection data outside the central region, shows that the reconstruction quality of the central square remains almost the same. This suggests that to reconstruct pixels in the

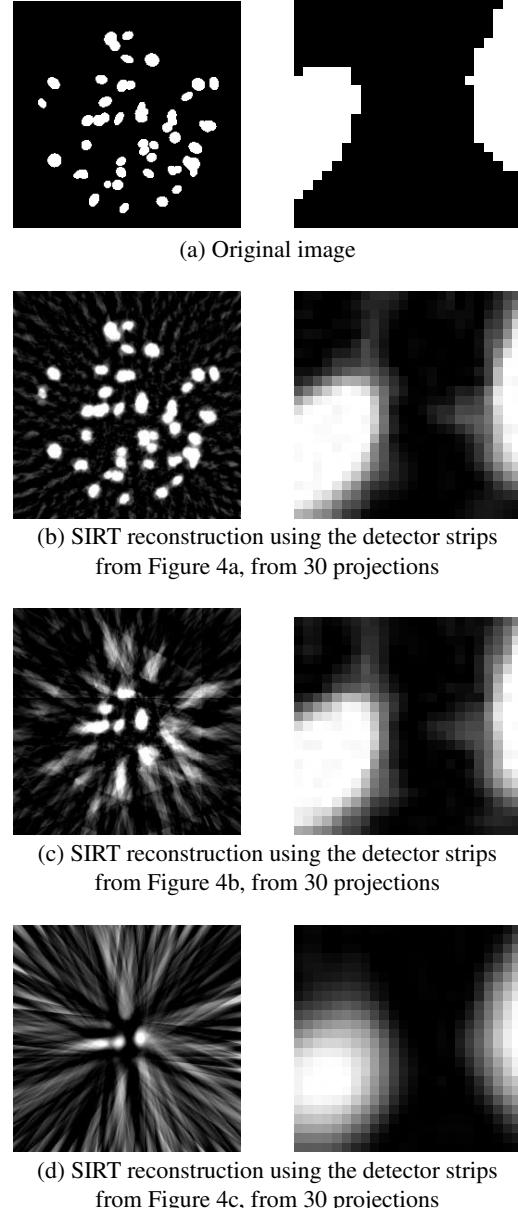


Figure 5. Left: A phantom image and three ART reconstructions, using the different sets of detectors strips from Figure 4. Right: Zoomed central region of the image on the left.

center of the image, knowing the most central part of each projection may already be sufficient. Figure 5d shows a SIRT reconstruction that was obtained by removing all strip projections outside the central region completely, corresponding to Figure 4c. We see that although the number of coarse strips in Figure 4b is fairly small, leaving them out of the reconstruction process completely results in an inferior reconstruction of the central square.

To reconstruct a single pixel instead of an entire region, we use even fewer strip projections. Starting from a fine strip that is centered on a given pixel p and the two direct neighbours of this strip, the width of the strips is doubled at each step further away from p . Using this exponential scheme, the number of strips for each projection angle is logarithmic in the number of detectors, vastly reducing

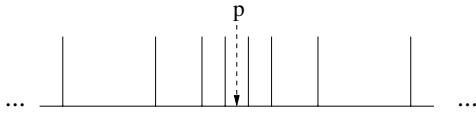


Figure 6. Starting from the projection of a pixel p , strips become exponentially wider as the distance from p increases.

the dimension of the input data for the classifier, see Figure 6. Note that the coarse detector strips do not correspond directly to physical detectors of the scanner. For each pixel, the strip projections for each of the corresponding coarse strips are *computed* from the measured projection data. By “compressing” the input data in this way, the number of inputs for the classification task is reduced from $O(dn)$ to $O(d \log n)$, where d denotes the number of projection angles and n denotes the number of detectors on the (physical) scanner. As we will see in Section 6, it is still possible to train a good classifier on the reduced input data.

4 FROM MANY CLASSIFIERS DOWN TO ONE

The main task for the classifier training algorithm consists of learning the mapping from the projection data to the value of a certain pixel. Clearly, the dependence of the pixel value on the projection data measured by the scanner is different for each pixel. However, after performing the compression step from the previous section, all input strip projections are *centered* around the position of the pixel. Therefore one would expect that a classifier that is trained for one pixel could also be used to compute the value of other pixels. The different dependencies on the measured strip projections are then addressed in the procedure that computes the input data for the classifier. The only remaining difference between the classification tasks for two different pixels is that the relative position of the imaging area A is different. If the set of training images contains some features that occur particularly often at specific positions in the image, the *position* of a pixel within the imaging area will be an important parameter in the classification task. To be able to model such location dependent information, two extra inputs are added to the set of input strip projection for the classifier, containing the x - and y -coordinate of the pixel. Figure 7 shows a schematic view of the resulting classification task.

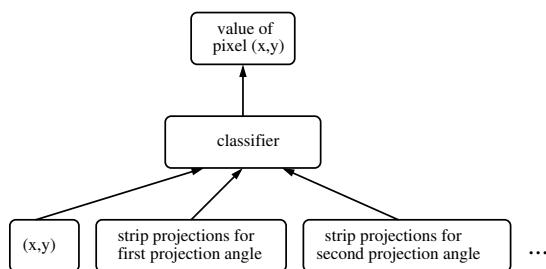


Figure 7. Schematic view of the classification task.

As the same classifier will be used to reconstruct each of the pixels, every training image (along with its projection data) can now be considered as an entire set of examples for the training algorithm: one for each pixel. In the experiments that are described in Section 6,

more than 10^7 examples are used to train the input-output relation. If a new image would be required for each new training example, the approach could not be used for practical applications at all, as it won't be possible to acquire so many scanner images. By using a single classifier for all pixels, each image that is acquired by the scanner yields thousands of training examples, which makes the approach practically feasible.

5 MORE THAN TWO GRAY VALUES

So far we have considered binary images, but the approach can also be used with more general images. For example, if all images in the training class contain three different gray values, the classifier may be trained to reconstruct the area covered by *one* of those gray values. The examples that are used by the training algorithm will still be binary images, that contain a 1 if a pixel has the desired gray value and a 0 otherwise. However, the projection data corresponds to the three-valued image. In that case, the classification that is learned by the training algorithm no longer corresponds to a conventional tomographic reconstruction. Using such a classifier, it may be possible to reconstruct an image of a single object in the tomographic scanner, while it is surrounded by various other objects. Doing this by conventional tomography algorithms would be an elaborate task: first the reconstruction that contains all objects has to be computed and subsequently the reconstruction has to be segmented as to separate the object of interest from the remaining part of the image. Section 6 presents several reconstruction results for images consisting of three gray values.

Clearly, our model can also be extended towards the full reconstruction of images that contain several gray values. The number of classes grows linearly with the number of gray values. The difficulty of the resulting training task is not only determined by the number of gray values and the types of images, but also by the difference between different gray values. Two gray values that have only a small difference will be difficult to distinguish.

6 RESULTS

Using compression of the input projection data of the classifier, as proposed in Section 3, reduces the input size significantly. Still, if more than just a few projection angles are used (i.e., more than 10) the input of the classifier may consist of more than 100 values. Tree-based classification models do not seem to be suitable for this task, as the number of features in the input space is too large. Both neural networks [1] and support vector machines [2] seem to be suitable for handling this large classification task. Efficient training algorithms are available that can deal with problems of this size. Neural networks for continuous tomography have already been proposed by several authors (see, e.g., [6]).

To demonstrate the feasibility of implementing tomographic reconstruction within a common learning classifier framework, we implemented a multilayer perceptron (MLP) with one hidden layer and backpropagation learning. All neuron activation functions are sigmoids. The network contains 50 hidden nodes and is trained for 200 epochs, each consisting of 250.000 training examples. Recall that a training example corresponds to a *pixel* of a *training image*, so each training image yields many different training examples. The MLP has one output, corresponding to the pixel value, which is thresholded at 0.5 to obtain a binary output. The training set consists of 1000 random binary images that have been formed by first drawing four white ellipses to create an “object” and subsequently draw-

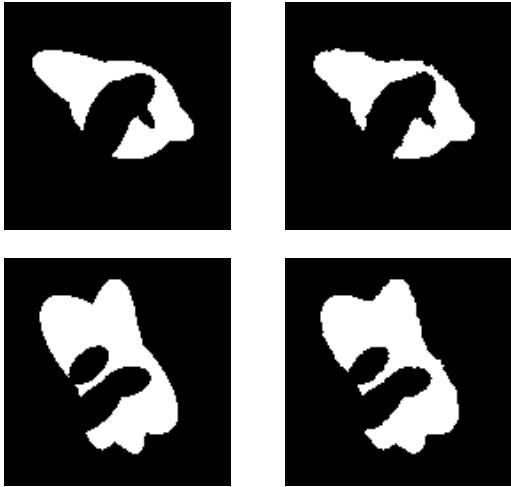


Figure 8. Left: two phantom images of size 128×128 . Right: reconstructions computed by the MLP classifier from 10 projections.

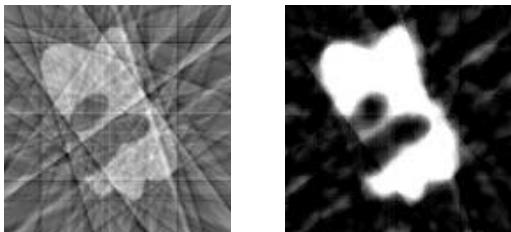


Figure 9. Left: Filtered Backprojection reconstruction of the top phantom in Figure 8 from 10 projections. Right: SIRT reconstruction from 10 projections.

ing three black ellipses inside the object to create various types of ‘holes’. The MLP was trained on images of size 128×128 . The total training time was around three hours. After training, the performance of the MLP was evaluated on a new set of 1000 random images that have a similar structure. Figure 8 shows two images from the test set and their reconstructions from 10 projections, equally spaced between 0 and 180 degrees. Averaged over all 1000 images in the test set, only 0.7% of all pixels are classified incorrectly by the MLP. Figure 9 shows reconstruction results from the same projection data by two continuous tomography algorithms. Filtered backprojection can be used in real-time, just as our MLP classifier, but it yields an inferior reconstruction. SIRT yields a much better reconstruction (though still not as good as the MLP), but on a standard 2.5GHz PC it takes about 20 seconds to run for a 128×128 image, compared to around 0.25s for the MLP (when used for all pixels in the image). By parallelisation of the MLP computations, even higher speeds are possible.

We also tested the MLP on a set of images that contain three pixel values (black, gray and white), training it to detect the presence of one of the pixel values (gray). Each image in the training and test sets consists of 10 ellipses that can be either gray or white, on a black background. Figure 10 demonstrates the performance of the trained network on a three-valued phantom image. Using 18 projections, the MLP is capable of detecting quite accurately where the gray pixels are directly from the projection data. For the test set of 1000 images, only 0.6% of all pixels are classified incorrectly by the MLP.

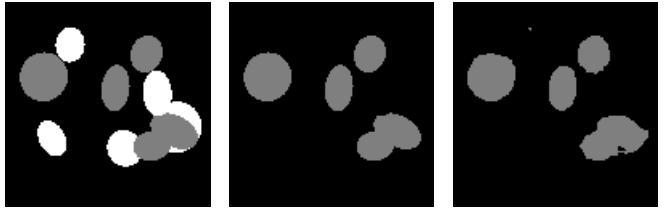


Figure 10. Left: Phantom image containing three gray levels. Middle: Gray part of the phantom image. Right: Reconstruction of the gray part, computed by the MLP from 18 projections of the three-valued phantom.

7 CONCLUSIONS

The goal of tomography is to reconstruct an object from its projections. If it is known beforehand that the object consists of only a few different materials (*discrete tomography*), the reconstruction problem can be considered as a *classification problem*. The dimension of the input space for the classifier is typically very large. We have described an approach for reducing the size of the input space in such a way that accurate classification is still possible using the reduced data. By defining all inputs *relative* to the position of a pixel, the same classifier can be used for all pixels in the image.

Our reconstruction results of an MLP classifier suggest that the reduced input data is indeed sufficient to perform accurate classification, at least for certain types of images. Once trained, the MLP can be used to compute new reconstructions very fast, and even more accurately than much slower algorithms from continuous tomography.

By training the classifier from a set of examples, prior knowledge of the set of images (besides the knowledge of the gray values) is learned automatically, instead of having to be modeled by hand.

Our approach is not limited to the reconstruction of binary images. In particular, it can be used to detect all pixels that have a certain gray value, directly from the projection data of an image that contains several gray values.

The experimental results were obtained using a training set of 1000 images. In many practical applications of tomography, 3D volumes are reconstructed that consist of several hundreds of stacked 2D slices. Therefore, a few 3D datasets would already be sufficient to perform training on real-world experimental data.

REFERENCES

- [1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
- [2] N. Cristianini and J. Shaw-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.
- [3] G. T. Herman and A. Kuba, eds., *Discrete Tomography: Foundations, Algorithms and Applications*, Birkhäuser, Boston, 1999.
- [4] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, SIAM, 2001.
- [5] N. Keat, ‘Real-time CT and CT Fluoroscopy’, *British Journal of Radiology*, **74**, 1088–1090, (2001).
- [6] J. P. Kerr and E. B. Bartlett, ‘Neural Network Reconstruction of Single-Photon Emission Computed Tomography Images’, *Journal of Digital Imaging*, **8**(3), 116–126, (1995).
- [7] P. A. Midgley, M. Weyland, J. M. Thomas and B.F.G. Johnson, ‘Z-Contrast Tomography: a Technique in Three-Dimensional Nanostructural Analysis Based on Rutherford Scattering’, *Chemical Communications*, **10**, 907–908, (2001).
- [8] F. Natterer, *The Mathematics of Computerized Tomography*, SIAM, 2001.

Depth Ordering and Figure-Ground Segregation in Monocular Images derived from Illusory Contour Perception

Marcus Hund and Bärbel Mertsching¹

Abstract. One of the elaborated tasks for the low level image processing stage of a vision system is the completion of cluttered or occluded object boundaries and the depth assignment of overlapped boundaries. We describe a novel method for depth ordering and figure-ground segregation from monocular depth cues, namely the arrangement of so-called illusory contours at junctions in the edge map of an image. Therefore, a computational approach to the perception of illusory contours, based on the tensor voting technique, is introduced. Furthermore, two simple rules for figure-ground segregation are presented that are capable of explaining a wide range of perceptual phenomena like the Kanizsa illusion. Finally, we present a diffusion approach to derive the depth ordering for the whole contour from the figure-ground segregation at key points.

1 Introduction

For technical as well as for biological vision systems, the completion of object boundaries that are interrupted due to occlusions or low luminance contrast is essential. This implies the distinction of completions of foreground object boundaries, called modal completions, and those of occluded background object boundaries, called amodal completions. The entirety of these completions are referred to as illusory contours. Illusory contours, also called virtual contours, are perceived contours that have no counterpart in the retinal image of the human vision system. Neurophysical studies have shown that the perception of illusory contours can be found in mammals, birds and insects [22]. The importance of illusory contours becomes obvious regarding the fact that the brains of these animals have developed independently throughout evolution. We can therefore assume that illusory contour perception is not just a malfunction of these visual processing systems, but instead is necessary for object border completion.

For the human vision system, illusory contours have been studied by Gestalt psychologists from the early 20th century on ([15], [3], [28]). Schuhmann was one of the first to mention this phenomenon in 1900 [27]. He described illusory contours as contours that are not "objectively present". In the following years the contributions to the field of illusory contour perception comprised the description of optical illusions based on contour perception rather than explaining these illusions.

Fig. 1(a) shows the Kanizsa triangle, an optical illusion which is the standard example for human mechanisms of illusory contour perception and for depth ordering derived from monocular inducers (see

also [13]). Even though there exists a multitude of correct scene interpretations leading to the same retinal image (Fig. 1(b)-(f)), the most often perceived depth ordering is the one shown in Fig. 1(b). Consequently, the aim of a computational approach to scene interpretation cannot be to find the correct depth ordering but to find the most probable scene interpretation.

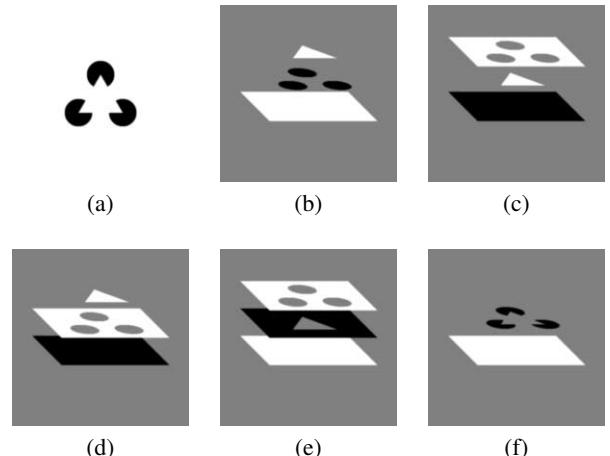


Figure 1. Some possible depth interpretations for the Kanizsa triangle shown in (a).

One possibility for figure-ground segregation is to consider contours that are interrupted by other contours. Such T-junctions can be interpreted as contours that are interrupted by a foreground object boundary and therefore belong to the background. But, as is stated in [10], where the figure-ground segregation was based on this principle, "such a simple mechanism can be fooled". For example, the stripes of a zebra produce T-junctions at the "object boundary", which would lead the mechanism to regard the zebra as the background. In such a case it is impossible to distinguish a hole from a textured object. In the following we will show that the ambiguities can be reduced if the possible completions of the interrupted contour are taken into account (compare Fig. 9).

2 Related Work

The majority of approaches or models dealing with the problem of spatial contour integration use some kind of bipole connection scheme ([7], [23]), as introduced by Grossberg and Mingolla [25].

¹ University of Paderborn, Germany, email: hund@get.uni-paderborn.de

This perceptual grouping kernel usually consists of two symmetric lobes encoding the connection strength and orientation. In [30], Williams and Thornber address the comparison of different methods of aggregating the contributions of neighboring sites of the grouping kernel. For a detailed overview of contour integration approaches, see [6] or [21]. In [21], emphasis is placed on models including illusory contour perception, namely the model of Heitger et al. ([9], [24]) and Neumann and coworkers [8] as a neurophysiological inspired computational model while the approach of Zweck and Williams [32] models the Brownian motion of a particle from source to sink.

The method proposed in this paper uses the tensor voting technique introduced by Medioni et al. [5]. Tensor voting was applied successfully to contour inference problems on synthetic and binary input images in [19]. In [16], this approach was extended to grayscale images as input, using gabor filtering as a preceding image processing step. In the tensor voting framework the grouping kernel, called stick voting field, is orientational, i.e. with angles from 0° to 180° and designed for contour inference. Considering illusory contour perception, the use of this stick voting field could make sense in the context of a unified treatment of all contour elements, but would lead to interference of contour elements, especially in the case of amodal completions behind a textured foreground. What is needed for illusory contours including amodal completions is a directional communication pattern (with angles from 0° to 360°), e.g. one lobe, which was already used in [18], but addressed to spontaneously splitting figures and binary input images.

Relatively little computational work has been done so far on figure-ground segregation derived from illusory contour cues. As mentioned above, Heitger et al. have implemented a merely rudimentary mechanism for figure-ground segregation [10]. In [14], the phenomena of unit formation and depth ordering are treated on a theoretical basis with focus on the neurophysiological view. Furthermore the "identity hypothesis" is proposed, stating that the mechanisms for modal and amodal completions are identical and that the depth ordering can be separated from the contour completion process. This view is challenged in [1], but the given arguments refer to examples using knowledge base or stereoscopic cues. Also no clues are given how the completion process takes place or how the depth ordering is realized. Regarding the monocular case in a low level processing step, we will give hints that, for this early stage, the depth ordering indeed can be realized with the contour completion being identical for modal and amodal completions. In this context unit formation has to be understood in the sense of contour formation.

3 Edge and Corner detection

As a preprocessing step we have developed a method for threshold-free edge detection and a subsequent corner detection. This is achieved by applying a recursive search to edge candidates. The edge candidates are local extrema and zero-crossings of the responses of several Gaussian-based filter banks. For an overview of edge detection algorithms, see [4] and [31].

The result of the recursive search is shown in Fig. 2(c), detected corners are shown superposed to the input image in Fig. 2(d). A comparison to other corner detectors is given in [20]. The used method for corner and edge detection is not an integral part of the proposed computational approach to illusory contour perception and can therefore be replaced by any other corner detector providing not only the corner positions but also the associated orientation angles. For a more detailed explanation of the used edge and corner detector, see [12].

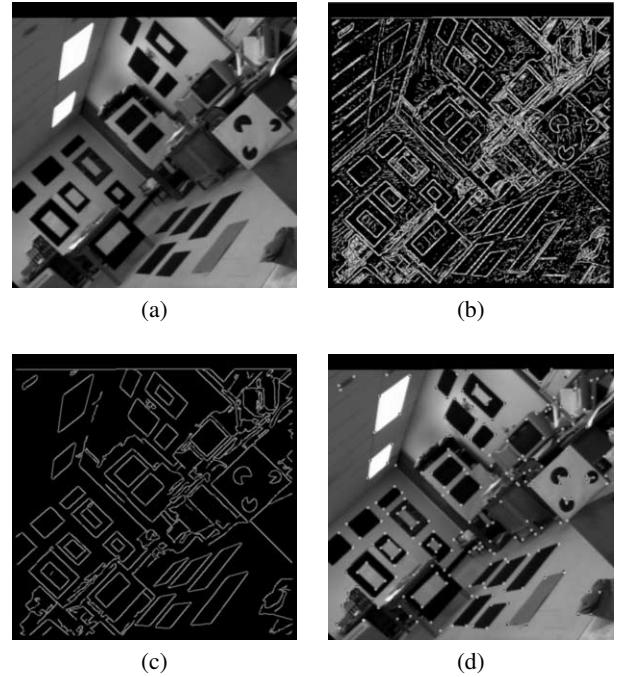


Figure 2. Lab scene: (a) Input image (b) Edge Candidates (c) Detected edges (d) Detected corners

4 Tensor Voting

In [19], Medioni, Lee and Tang describe a framework for feature inference from sparse and noisy data called tensor voting. The most important issue is the representation of edge elements as tensors. In the 2D-case, a tensor over \mathbb{R}^2 can be denoted by a symmetric 2×2 matrix T with two perpendicular eigenvectors \vec{e}_1, \vec{e}_2 and two corresponding real eigenvalues $\lambda_1 > \lambda_2$. A tensor can be visualized as an ellipse in 2-D with the major axis representing the estimated tangent direction \vec{e}_1 and its length λ_1 reflecting the saliency of this estimation. The length λ_2 assigned to the perpendicular eigenvector \vec{e}_2 encodes the orientation uncertainty. The definition of saliency measures is deducted from the following decomposition of a tensor into $T = \lambda_1 \vec{e}_1 \vec{e}_1^\top + \lambda_2 \vec{e}_2 \vec{e}_2^\top$ or equivalently $T = (\lambda_1 - \lambda_2) \vec{e}_1 \vec{e}_1^\top + \lambda_2 (\vec{e}_1 \vec{e}_1^\top + \vec{e}_2 \vec{e}_2^\top)$. Then, the weighting factor $(\lambda_1 - \lambda_2)$ represents an orientation in the direction of the eigenvector \vec{e}_1 and thus will be called *curve- or stick-saliency*. The second weight λ_2 is applied to a circle, hence it is called *junction- or ball-saliency* as its information about multiple orientations measures the confidence in the presence of a junction. Note that for junctions or corners neither the tensor representation suffices to encode the at least two different orientations nor is the ball saliency a trustable measure for junctions, since it is highly dependent on the orientations of incoming edges [17].

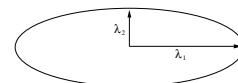


Figure 3. Visualization of a tensor as an ellipse

Grouping can now be formulated as the combination of elements according to their stick-saliency or ball-saliency. In stick-voting, for each oriented input token the grouping kernel called stick-voting-

field is aligned to the eigenvector \vec{e}_1 . In the following the input tokens consist of detected junctions and their associated directions. All fields are combined by tensor addition, i.e. addition of the matrices and spectral decomposition of the sum into eigenvectors and -values. The field is designed to create groupings with neighboring tokens which fulfill the minimal curvature constraint. Hence the orientation of each token of the voting field is defined to lie on a cocircular path. The strength is defined as follows:

Given a point P with an associated tangent direction and a point Q with the orientation difference θ between the tangent direction and the direct connection of P and Q . Let ℓ be the distance between P and Q . Then, with

$$r = \frac{\ell}{2\sin\theta} \quad \text{and} \quad s = \frac{\ell \cdot \theta}{\sin\theta} \quad ,$$

r is the radius of the tangent circle to P going through Q and s is the arc length distance along the circular path (radian).

Most approaches to spatial contour integration define the connection strength V for P and Q and therefore the shape of the bipole connection scheme via $V = V_d \cdot V_c$ with a distance term V_d and a curvature term V_c . In [9], Heitger et al. use

$$V_{d1} = e^{-\frac{\ell^2}{2\sigma^2}} \quad \text{and} \quad V_{c1} = \begin{cases} \cos^k(\frac{\pi/2}{\alpha} \cdot \theta) & \text{if } |\theta| < \alpha \\ 0 & \text{otherwise} \end{cases}$$

with $k = 2n$, $n \in \mathbb{N}$ and an opening angle $2\alpha = \pi$. Hansen and Neumann also use V_{d1} and V_{c1} , but with $k = 1$ and $\alpha = 10^\circ$ [7]. In [19], Medioni et al. define the proximity term V_{d2} and the curvature term V_{c2} as follows:

$$V_{d2} = e^{-\frac{s^2}{2\sigma^2}} \quad \text{and} \quad V_{c2} = e^{-\frac{c \cdot \rho^2}{\sigma^2}} \quad \text{with} \quad \rho = \frac{2\sin\theta}{\ell}$$

c is a positive constant and ρ is nothing else than the inverse radius of the osculating circle. This results in a curvature measure that is highly dependent on scale.



Figure 4. Stick saliency for the half lobe stick voting field with $V = V_{d2} \cdot V_{c1}$, $k = 1$ and $\alpha = 15^\circ$

To achieve a clear separation of distance along the circular path and its curvature, we choose V_{d2} and V_{c1} . The results presented in the next section are computed with a directional one-lobe voting field and $V = V_{d2} \cdot V_{c1}$, $k = 1$ and $\alpha = 15^\circ$, i.e. an opening angle of 30° .

Starting from junctions the illusory contour perception is now realized by letting the junctions cast Stick Voting Fields according to their associated orientations. By a maximum search along the voting field we try to find a connection to other junctions.

5 Illusory Contour Perception

Fig. 5(a) shows detected edges and corners and their associated orientations, (b) shows the stick saliences after tensor voting and (c) shows extracted illusory contours superposed to the previously detected contours. It is remarkable, that in Fig. 5(c) the amodal completions of the circles are not found. This is due to the acute connection angles as can be seen in Fig. 5(b).

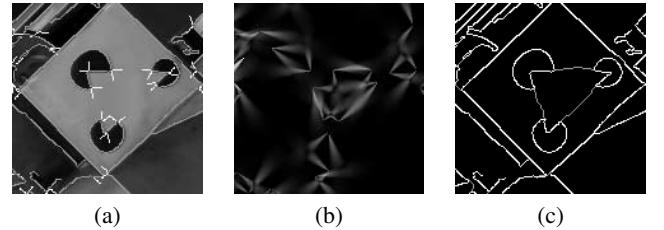


Figure 5. Illusory contour perception for the Kanizsa triangle in Fig. 2. For further description see text.

In Fig. 6, the rubber duck is partially occluded by a black cable. Note that there are some light reflections on the cable causing a foreground texture. For contour inference approaches, i.e. the application of bipole voting fields to all edgels, the voting fields cast by these foreground elements would interfere with the fields generated at the duck's object boundary and hence compromise the correct detection of amodal completions (Fig. 6(b)).

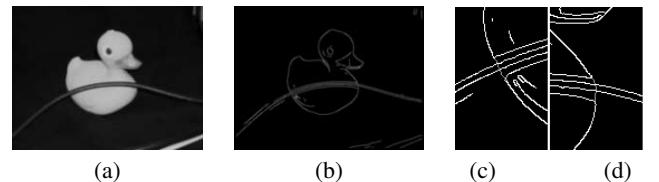


Figure 6. Rubber duck image: (a) input image, (b) illusory contours superposed to detected edges, (c) and (d) magnification of regions containing illusory contours

This illustrates that a unified treatment of edge segments and junctions would disturb the perception of amodal completions, at least for this low level image processing step. Anyhow just the two desired amodal completions of the duck's object boundary are marked as illusory contours, see Fig. 6(d) and (e), so the correct virtual contours are found.

6 Figure-Ground Segregation and Depth Ordering

The figure-ground segregation for the results shown in the next section is based on two simple rules for the crossing of two contours:

If a contour changes its state from real to illusory at a point of interest and if a crossing contour does not, then the first contour belongs to the background and the second contour belongs to an occluding foreground surface.

If a contour changes its state from real to illusory and if the illusory part lies within the bounded region of a closed crossing contour, then the first contour belongs to the background and the second contour is the closed boundary of an occluding object. If both contours are closed and both contours have illusory parts that lie within the bounded region of the other contour, no reasonable decision can be made. This would be the case of spontaneously splitting figures.

More complex graph based labeling schemes can be found in [29] and [26], but these two rules already suffice to interpret a variety of situations in an accurate way, e.g. the second rule is sufficient to describe the complex depth ordering in the Kanizsa triangle. While the

second rule describes a more complex scene understanding mechanism, the first rule is even simpler and just requires the treatment of local image structure.

To ensure that the depth assignment is continuous even for contours that bound surfaces that change from back- to foreground and vice versa, we realize the depth ordering with a diffusion process: Let x_n be the relative depth of a contour element with neighbors $x_{n+1}, x_{n+2}, x_{n-1}$ and x_{n-2} . Depth of a contour means the depth of the occluding surface if the contour marks the boundary of a surface. Given a smoothing factor a , the cost term

$$\begin{aligned} C = \frac{1}{2} \sum_n & a \cdot (((x_{n+1} - x_n) - (x_n - x_{n-1}))^2 \\ & + (x_n - x_{n+1})^2 + (x_n - x_{n-1})^2) \\ & + \begin{cases} (x_n - x_0)^2 & \text{if } x_n \text{ belongs to a junction} \\ 0 & \text{otherwise} \end{cases} \quad (1) \end{aligned}$$

enforces the smoothness of neighboring values as well as of the first derivative. Using the gradient descent method to minimize Equation (1) leads to

$$\begin{aligned} -\frac{\partial}{\partial x_n} C = & -a \cdot (6x_n - 4x_{n+1} - 4x_{n-1} + x_{n+2} + x_{n-2} \\ & + 4x_n - 2x_{n+1} - 2x_{n-1}) \\ & - \begin{cases} 2(x_n - x_0) & \text{if } x_n \text{ belongs to a junction} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Then, with λ being the step width, the update rule for x_n is given by

$$x_n^{(k+1)} = x_n^{(k)} - \lambda \cdot \frac{\partial}{\partial x_n} C \quad (2)$$

During the dynamic process we update the x_0 values at points of interest according to the figure-ground segregation described above.

7 Results

Fig. 7 illustrates the different processing steps to the given input images in 7(a). Fig. 7(b)-(e) shows detected edges and the possible orientations at junctions. In (c) the votes cast by these junctions are shown and in (d) the resulting illusory contours are superposed to the real contours.

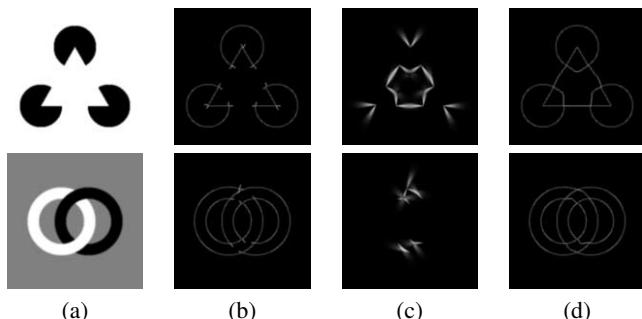


Figure 7. Kanizsa Triangle (top row) and Rings (bottom row): (a) Input Images , (b)-(e) Different processing steps. For further description see text.

In Fig. 8 and the bottom row of Fig. 9 the results for the figure-ground and depth ordering mechanisms are shown. The contour elements bounding a foreground surface are displayed lighter while edges being assigned a higher distance to the viewer are displayed darker.

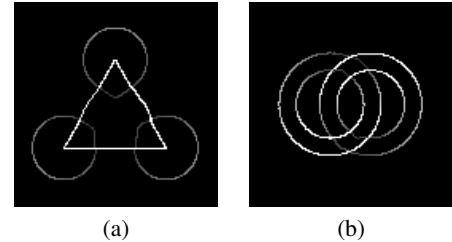


Figure 8. Depth ordering for the images in fig. 7(a)

Fig. 8(a) shows that the second rule presented in the previous section lets the mechanism interpret the depth ordering that is identical to human perception. In 8(b) the figure-ground segregation results from the first rule. As can be seen, eq. 2 ensures that even with key points changing from figure to ground a continuous depth signal can be produced.

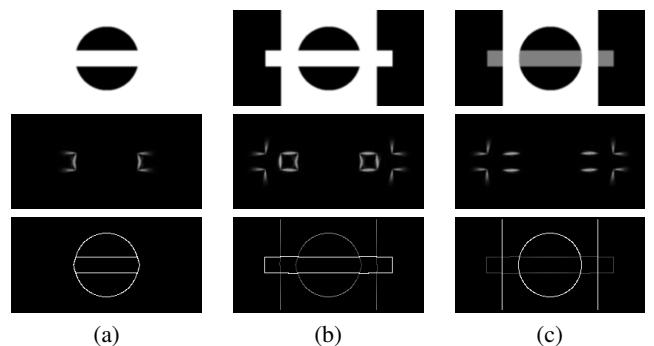


Figure 9. Top row: Input images for a circle with a bar, middle row: Voting fields cast by detected junctions, bottom row: Real and illusory contours with assigned depth values. For further description see text.

Fig. 9(a) shows the "zebra case". The constellation is highly ambiguous, since there is no cue to decide whether the white bar in the black circle is foreground, seen through a hole or part of the circle's surface texture. In Fig. 9(b) the last case, being part of the circle's surface, is eliminated by the fact that the bar is bounded by a closed contour. The second rule presented in the previous section enforces the perception of a bar in front of the circle, as can be seen in the bottom row, which correlates with the perception of the majority of observers. Also most observers perceive a white surface containing a hole in front of a grey bar in front of a black background in Fig. (9(c)). The bottom row shows that the first rule presented in the previous section leads exactly to this depth ordering.

8 Conclusion

A computational approach was presented for the depth ordering at surface boundaries in monocular images. The approach includes corner- and edge detection, illusory contour perception, figure-ground segregation at junctions and depth ordering for real and completed contours. The illusory contour perception is based on the Tensor Voting framework, for what we constructed a new communication pattern that allows a clear distinction between the distance along a circular path and its curvature. The figure-ground segregation is done by the application of two simple rules, one regarding local image content and one placing emphasis on closed contours. Furthermore we invented a diffusion process, in which the figure-ground

segregation at key points is carried to the rest of the contour. It was shown that this proceeding even works with two surfaces occluding one another. Consequently, this approach allows an assumption about an object's relative position in space, not just a distinction between figure and ground. The promising results presented for these mechanisms may be considered as an evidence for Kellmanns "identity hypothesis" [14], but on the other hand, regarding a larger scale, it seems clear that contour perception is also influenced by top down processes driven by knowledge base.

9 Future work

Obviously, binocular depth cues belong to the most significant inducers for depth perception. Our future work will include a fusion of work we have done on stereoscopic algorithms ([2],[11]) with the mechanisms presented here. On the one hand, the depth ordering presented here can be used as an input to stereoscopic disparity estimation, on the other hand the results of a stereoscopic algorithm can be feeded back to the figure-ground segregation step. Furthermore, all this can be realized without any knowledge about the objects in the world that is observed.

REFERENCES

- [1] B. L. Anderson, M Singh, and R. W. Fleming, 'The interpolation of object and surface structure.', *Cognitive Psychology*, **44**, 148–190, (2002).
- [2] Roland Brockers, Marcus Hund, and Baerbel Mertsching, 'Stereo Matching with Occlusion Detection Using Cost Relaxation', in *IEEE International Conference on Image Processing*, (September 2005).
- [3] Walter H. Ehrenstein, Lothar Spillmann, and Viktor Sarris, 'Gestalt issues in modern neuroscience', *Axiomathes*, **13**(3), 433–458, (2003).
- [4] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Prentice Hall, 2002.
- [5] Gideon Guy and Gerard Medioni, 'Inferring global perceptual contours from local features', *International Journal of Computer Vision*, **20**(1-2), 113–133, (1996).
- [6] Thorsten Hansen, *A neural model of early vision: Contrast, contours, corners and surfaces*, Ph.D. dissertation, Universität Ulm, 2003.
- [7] Thorsten Hansen and Heiko Neumann, 'Neural mechanisms for representing surface and contour features', in *Emergent Neural Computational Architectures Based on Neuroscience - Towards Neuroscience-Inspired Computing*, pp. 139–153. Springer-Verlag, (2001).
- [8] Thorsten Hansen, Wolfgang Sepp, and Heiko Neumann, 'Recurrent long-range interactions in early vision', **2036**, 127, (2001).
- [9] F. Heitger and R. von der Heydt, 'A computational model of neural contour processing: Figure-ground segregation and illusory contours', in *International Conference on Computer Vision*, pp. 32–40, (1993).
- [10] Friedrich Heitger, Rüdiger von der Heydt, Esther Peterhans, Lukas Rosenthaler, and Olaf Kübler, 'Simulation of neural contour mechanisms: representing anomalous contours.', *Image Vision Comput.*, **16**(6-7), 407–421, (1998).
- [11] Marcus Hund, *Disparitätsbestimmung aus Stereobildern auf der Basis von Kostenfunktionen*, diploma thesis, University of Paderborn, August 2002.
- [12] Marcus Hund and Baerbel Mertsching, 'A Computational Approach to Illusory Contour Perception Based on the Tensor Voting Technique', in *10th Ibero-American Congress on Pattern Recognition (XCIARP)*, (November 2005).
- [13] *Organization in Vision*, ed., G. Kanizsa, Praeger, 1979.
- [14] P.J. Kellman, S. E. Guttman, and T. D. Wickens, 'Geometric and neural models of object perception', in *From fragments to objects: Segmentation and grouping in vision*. T. F. Shipley, Ed, and P.J. Kellman, Ed. . Oxford, UK: Elsevier Science, (2001).
- [15] K. Koffka, *Principles of Gestalt psychology*, Harcourt Brace, New York, 1935.
- [16] A. Massad, M. Babos, and B. Mertsching, 'Perceptual grouping in grey level images by combination of gabor filtering and tensor voting', in *ICPR*, eds., R. Kasturi, D. Laurendeau, and C. Suen, volume 2, pp. 677–680, (2002).
- [17] A. Massad, M. Babos, and B. Mertsching, 'Application of the tensor voting technique for perceptual grouping to grey-level images: Quantitative evaluation', (2003). Intl. Symposium on Image and Signal Processing and Analysis.
- [18] A. Massad and G. Medioni, '2-D Shape Decomposition into Overlapping Parts', in *Visual Form 2001, 4th International Workshop on Visual Form (IWVF 4)*, eds., C. Arcelli, L. Cordella, and G. Sanniti di Baja, pp. 398 – 409, Capri, Italy, (January 2001).
- [19] G. Medioni, M.-S. Lee, and C.-K. Tang, *A Computational Framework for Segmentation and Grouping*, Elsevier Science, 2000.
- [20] Farzin Mokhtarian and Riku Suomela, 'Robust image corner detection through curvature scale space', *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**(12), 1376–1381, (1998).
- [21] H. Neumann and E. Mingolla, 'Computational neural models of spatial integration in perceptual grouping', in *From fragments to units: Segmentation and grouping in vision*, eds., T. Shipley and P. Kellman, 353–400, Elsevier Science, Oxford, UK, (2001).
- [22] Andreas Nieder, 'Seeing more than meets the eye: processing of illusory contours in animals', *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, **188**(4), 249–260, (2002).
- [23] Pierre Parent and Steven Zucker, 'Trace inference, curvature consistency, and curve detection', *IEEE Trans. Pattern Anal. Mach. Intell.*, **11**(8), 823–839, (1989).
- [24] Esther Peterhans and Friedrich Heitger, 'Simulation of neuronal responses defining depth order and contrast polarity at illusory contours in monkey area v2.', *Journal of Computational Neuroscience*, **10**(2), 195–211, (2001).
- [25] W. D. Ross, S. Grossberg, and E. Mingolla, 'Visual cortical mechanisms of perceptual grouping: interacting layers, networks, columns, and maps', *Neural Netw.*, **13**(6), 571–588, (2000).
- [26] Eric Saund, 'Perceptual organization of occluding contours of opaque surfaces', *Comput. Vis. Image Underst.*, **76**(1), 70–82, (1999).
- [27] F. Schumann, 'Beiträge zur Analyse der Gesichtswahrnehmungen. Erste Abhandlung. Einige Beobachtungen über die Zusammenfassung von Gesichtseindrücken zu Einheiten.', *Zeitschrift für Psychologie und Physiologie der Sinnesorgane*, **23**, 1–32, (1900). English translation by A. Hogg (1987) in *The perception of Illusory Contours* Eds S Petry, G.E. Meyer (New York: Springer) pp 40-49.
- [28] Max Wertheimer, 'Untersuchungen zur Lehre von der Gestalt II', *Psychologische Forschung*, **4**, 301–350, (1923).
- [29] Lance R. Williams and Allen R. Hanson, 'Perceptual completion of occluded surfaces', *Computer Vision and Image Understanding: CVIU*, **64**(1), 1–20, (1996).
- [30] Lance R. Williams and Karvel K. Thornber, 'A comparison of measures for detecting natural shapes in cluttered backgrounds', *International Journal of Computer Vision*, **34**(2/3), 81–96, (2000).
- [31] D. Ziou and S. Tabbone, 'Edge detection techniques: an overview', *International Journal on Pattern Recognition and Image Analysis*, **8**(4), 537–559, (1998).
- [32] John W. Zweck and Lance R. Williams, 'Euclidean group invariant computation of stochastic completion fields using shiftable-twistable functions.', in *ECCV* (2), pp. 100–116, (2000).

Graph Neural Networks for Object Localization

Gabriele Monfardini and Vincenzo Di Massa and Franco Scarselli and Marco Gori¹

Abstract. Graph Neural Networks (GNNs) are a recently proposed connectionist model that extends previous neural methods to structured domains. GNNs can be applied on datasets that contain very general types of graphs and, under mild hypotheses, they have been proven to be universal approximators on graphical domains. Whereas most of the common approaches to graphs processing are based on a preliminary phase that maps each graph onto a simpler data type, like a vector or a sequence of reals, GNNs have the ability to directly process input graphs, thus embedding their connectivity into the processing scheme. In this paper, the main theoretical properties of GNNs are briefly reviewed and they are proposed as a tool for object localization. An experimentation has been carried out on the task of locating the face of a popular Walt Disney character in comic covers. In the dataset the character is shown in a number of different poses, often in cluttered backgrounds, and in high variety of colors. The proposed learning framework provides a way to deal with complex data arising from image segmentation process, without exploiting any prior knowledge on the dataset. The results are very encouraging, prove the viability of the method and the effectiveness of the structural representation of images.

1 Introduction

Object localization [4, 3], image classification [7], natural language processing [14, 6], bioinformatic [2], Web page scoring, social networks analysis [16] and relational learning are examples of machine learning applications where the information of interest are encoded into the relationships between a set of basic entities. In those domains, data is suitably represented as sequences, trees, and, more generally, directed or undirected graphs. In fact, nodes can be naturally used to denote concepts with edges specifying their relationships. For instance, an image can be represented by a Region Adjacency Graph (RAG), where the nodes denote the homogeneous regions of the image and the edges represent their adjacency relationship (Fig. 1).

In those applications, the goal consists of designing a function τ that maps a graph G and one of its nodes n to a vector of reals $\tau(G, n)$. For example, the localization of an object in a image can be implemented by a function τ that classifies the nodes of the RAG according to whether the corresponding region belongs to the object or not. In the case of Fig. 1, where the object to be localized is the plane, $\tau(G, n)$ might be 1 for the black nodes, which correspond to the parts of the plane, and -1 otherwise.

Traditional methods usually deal with graphs using a preprocessing procedure that transforms the graphs into simpler representations, as vectors or sequences of reals, which are then elaborated with

common machine learning techniques. However, valuable information may be lost during the preprocessing and, as a consequence, the application may suffer from a poor performance and generalization.

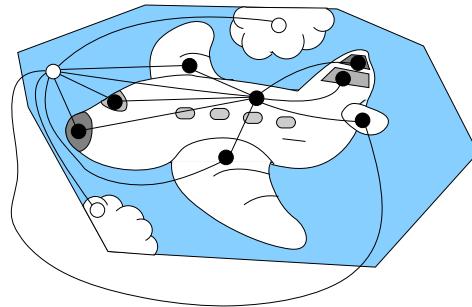


Figure 1. An image and its graphical representation by a RAG. Black nodes are associated to the parts of the plane, white nodes to the background.

Recently, a new connectionist model, called Graph Neural Network (GNN) model, has been proposed to better exploit the graphical representation of data [9, 20]. In GNNs, the preprocessing procedure is not required since the relationships between the basic entities of the input domain are directly embedded into the processing scheme of the model. It has been proved that, under mild hypotheses, GNNs are universal approximators on graphical domains, i.e. the can approximate any function $\tau(G, n)$.

GNNs extend the previous connectionist methods for structured domains [11] like Recursive Neural Networks (RNNs) [21, 8], and SOM for structured domains (SOM-SD) [10]. Actually, whereas RNNs and SOM-SD were limited by the fact that they can be applied only on a restricted class of structures, i.e. the directed acyclic graphs, GNNs can virtually elaborate any type of graphs, including cyclic and acyclic, directed and undirected ones. Moreover, the original RNN model produces one output for each graph and can be applied only on those problems where $\tau(G, n)$ does not depend on the node n . Thus, for example, RNNs are suitable for image classification, but not for object localization².

However GNNs had not been widely validated experimentally since now and it was unknown whether they could be applied with success on practical problems. In this paper, the results on an object localization problem are presented. The task consists of locating the face of a popular Walt Disney character in comic covers. Each cover was segmented and represented by a RAG, with labeled nodes and edges.

¹ Università degli Studi di Siena, Dip. Ingegneria dell'Informazione, Italy,
 email: {monfardini,dimassa,franco,marco}@dii.unisi.it

² More precisely, object localization in images can be carried out even by RNNs, but, in this case, the input RAGs must undergo a preprocessing step that transforms RAGs to direct acyclic graphs [4] (see Sect. 3).

In comic covers the character is shown in a number of different poses, often in cluttered backgrounds, and in high variety of colors. The proposed learning framework provides a way to deal with complex data arising from image segmentation process, without exploiting any prior knowledge on the dataset.

Moreover, common approaches to object localization are usually based on a window which is moved over the image [23, 22]. At each step of such a procedure, a classification technique is used to verify whether the window currently contains the target object. On the other hand, GNNs process the whole image representation once and for all, exploiting the graphical structure encoded naturally in the RAG.

The paper is organized as follows. In the next section, the GNN model and its main properties are reviewed. Section 3 illustrates the object localization problem and shows the results obtained by GNNs. Finally, the conclusions are drawn in Sect. 4.

2 Graph Neural Networks

In the following, a *graph* G is a pair (N, E) , where N is a set of *nodes* (or vertices), and E is a set of *edges* (or arcs) between nodes: $E \subseteq \{(u, v) | u, v \in N\}$. If the graph is *directed*, each edge (u, v) has a direction, i.e. a head and a tail, whereas, in *undirected* graphs, the ordering between u and v is not defined, i.e. $(u, v) = (v, u)$. Nodes and edges may be labeled and labels are represented by l_n and $l_{(u,v)}$, respectively. By convention, the symbol l with no further specifications defines all the labels of the graph, while operator $| \cdot |$ denotes the cardinality or the absolute value of its argument, according to whether it is applied to a set or to a number. A graph is called *acyclic* if there is no path, i.e. a sequence of connected edges, that starts and ends in the same node, otherwise it is *cyclic*. Finally, the nodes adjacent to some node n (or neighbors of n) are those connected to it by an edge and are represented by $\text{ne}[n]$.

2.1 The GNN model

The GNN model has been conceived to approximate a target function τ that maps a graph G and one of its nodes n into an output vector $o_n = \tau(G, n)$. In categorization problems $o_n \in \mathbb{N}^m$, while in regression problems the outputs belong to \mathbb{R}^m .

In GNN framework, a node represents an object or a concept of the domain of interest. Each node can be described using a fixed-length vector of real numbers $x_n \in \mathbb{R}^s$ called *state*, where the dimension s is a predefined parameter of the model. In order to derive a flat but adaptive description of each node n in a distributed processing scheme, the state must be evaluated locally at each node. The goal here is to combine label symbolic information with subsymbolic contributions embedded in graph topology. Such a task is accomplished modeling x_n as the output of a parametric function f_w (called *state transition function*), that depends on the node label l_n and on the relationships between n and its neighbors

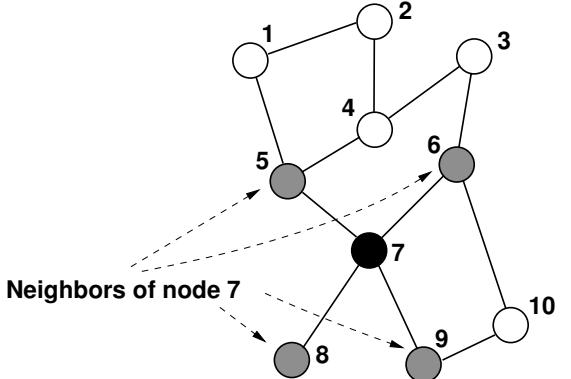
$$x_n = f_w(l_n, x_{\text{ne}[n]}, l_{\text{ne}[n]}, l_{(n, \text{ne}[n])}), \quad n \in N, \quad (1)$$

where $\text{ne}[n]$ is the set of neighbors of node n . Here, $x_{\text{ne}[n]}$ and $l_{\text{ne}[n]}$ are the states and the labels of the nodes in $\text{ne}[n]$, respectively, and $l_{(n, \text{ne}[n])}$ collects the labels of the edges connected to n (Fig. 2).

For each node n , the state x_n is also combined with node label l_n to produce an output vector o_n

$$o_n = g_w(x_n, l_n), \quad n \in N, \quad (2)$$

where g_w is another parametric function, called *output function*.



$$\begin{aligned} x_7 = f_w(l_7, x_5, x_6, x_8, x_9, l_5, l_6, l_8, l_9, \\ l_{(7,5)}, l_{(7,6)}, l_{(7,8)}, l_{(7,9)}) \end{aligned}$$

Figure 2. The state x_n depends on the information in its neighborhood.

In conclusion, Eqs. (1) and (2) define a method to compute an output $o_n = \varphi_w(G, n)$ for each node n of the graph G , taking into account the local descriptions and the relationships of all the nodes in G . The goal of the learning procedure is to adapt the parameters w so that φ_w approximates the data in the learning set $\mathcal{L} = \{(G_i, n_{i,j}, t_{n_{i,j}}) | 1 \leq i \leq p, 1 \leq j \leq q_i\}$, where each triple $(G_i, n_{i,j}, t_{n_{i,j}})$ denotes a graph G_i , one of its nodes $n_{i,j}$ and the desired output $t_{n_{i,j}}$. Moreover, p is the number of graphs in \mathcal{L} and q_i is the cardinality of the set of supervised nodes in graph G_i . In practice, the learning problem can be implemented as the minimization of the quadratic error function

$$e_w = \sum_{i=1}^p \sum_{j=1}^{q_i} (t_{n_{i,j}} - \varphi_w(G_i, n_{i,j}))^2. \quad (3)$$

Notice, however, that the following three issues must be solved in order to implement the proposed model.

1. How can the transition function f_w and the output function g_w be implemented?
2. The state x_n should be uniquely defined, but Eq. (1) provides a recursive specification of x_n . Thus, how can be ensured that Eq. (1) has a unique solution?
3. How can the parameters be adapted to minimize the error function?

In GNNs, both f_w and g_w can be implemented by two feedforward neural networks. However, the fact that f_w doesn't have a fixed number of input parameters, as different nodes have different numbers of neighbors (see Eq (1)), raises some practical problems. In fact, the network implementing f_w should be designed to accommodate the maximal number of input parameters, even if several of them may remain unused in most of the cases. A better solution consists of specializing Eq. (1)

$$x_n = \sum_{i=1}^{|\text{ne}[n]|} h_w(l_n, x_{\text{ne}_i[n]}, l_{\text{ne}_i[n]}, l_{(n, \text{ne}_i[n])}), \quad n \in N \quad (4)$$

where $\text{ne}_i[n]$ is the i -th neighbor of n , $|\text{ne}[n]|$ is the number of neighbors of n , and h_w is parametric function implemented by a feedfor-

ward neural network. Here, the idea is that \mathbf{x}_n can be evaluated as a sum of “contributions”, one for each of the neighbors of node n .

In order to solve the second issue, notice the set of Eqs. (1) can be also represented as

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} f_w(l_1, \mathbf{x}_{ne[1]}, l_{ne[1]}, l_{(1,ne[1])}) \\ f_w(l_2, \mathbf{x}_{ne[2]}, l_{ne[2]}, l_{(2,ne[2])}) \\ \vdots \\ f_w(l_N, \mathbf{x}_{ne[N]}, l_{ne[N]}, l_{(N,ne[N])}) \end{bmatrix}$$

where N is the cardinality $|N|$ of node set N . Introducing a vectorial function \mathbf{F}_w obtained stacking together the instances of the transition function f_w , collecting all the states in the symbol \mathbf{x} and all the labels in the symbol \mathbf{l} , the previous equation can be rewritten as

$$\mathbf{x} = \mathbf{F}_w(\mathbf{x}, \mathbf{l}). \quad (5)$$

By Banach Theorem, if \mathbf{F}_w is a contraction mapping³, then Eq. (5) has a unique solution [13]. Thus, the second issue can be solved by designing f_w such that the global function \mathbf{F}_w results to be a *contraction mapping* w.r.t. the state \mathbf{x} .

In practice, this goal can be achieved by adding a penalty term to the error function

$$e_w = \sum_{i=1}^p \sum_{j=1}^{q_i} (\mathbf{t}_{n_{i,j}} - \varphi_w(\mathbf{G}_i, n_{i,j}))^2 + \beta L \left(\left\| \frac{\partial \mathbf{F}_w}{\partial \mathbf{x}} \right\| \right),$$

where $L(y) = (y - \mu)^2$, if $y > \mu$, and $L(y) = 0$ otherwise [20]. Moreover, the real number β balances the importance of the penalty term and the error on patterns, the parameter $\mu \in (0, 1)$ defines a desired upper bound on the contraction constant of \mathbf{F}_w , and $\left\| \frac{\partial \mathbf{F}_w}{\partial \mathbf{x}} \right\|$ denotes the norm of the Jacobian of \mathbf{F}_w .

Next section answers briefly to the last issue, discussing the learning algorithm used to adapt model parameters. More details can be found in [20].

2.2 The learning algorithm

The functions h_w and g_w may be implemented by a variety of feedforward neural network architectures. In our experiments, they are designed as three layered (i.e. with one hidden layer) feedforward neural networks. According to Banach fixed point theorem, if ρ is a generic contraction mapping, then the dynamical system $\mathbf{x}(t+1) = \rho(\mathbf{x}(t))$, where $\mathbf{x}(t)$ denotes the t -th iterate of \mathbf{x} , converges exponentially fast to the solution of the equation $\mathbf{x} = \rho(\mathbf{x})$ for any initial state $\mathbf{x}(0)$. Thus Eq. (5) can be solved simply by iteration. Then, fixed point states are used by function g_w to evaluate the outputs \mathbf{o}_n (Eq. (2)), while the weight updating is accomplished deriving the error according to Eq. (3) and employing a gradient descent strategy as in traditional backpropagation algorithm.

The main difference with respect to the traditional backpropagation is due to the fact that fixed point states depend on the parameters of f_w through the iteration procedure. For this reason the error should be backpropagated from one iteration to the previous, accumulating the gradient until convergence. The adopted gradient computation is not completely new: it combines the backpropagation through structure algorithm, designed to train Recursive Neural Networks [21, 8], and the recurrent backpropagation algorithm [1, 18].

³ A generic function $\rho : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a contraction mapping w.r.t. a vector norm $\|\cdot\|$, if it exists a real number μ , $0 \leq \mu < 1$, such that for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, $\|\rho(\mathbf{x}_1) - \rho(\mathbf{x}_2)\| \leq \mu \|\mathbf{x}_1 - \mathbf{x}_2\|$.

When the gradient is available, the weights can be updated using resilient backpropagation [19], a well-known variant of standard backpropagation that speeds up the convergence.

3 Application to object localization

In order to test the performance of the Graph Neural Network model, an object localization problem was designed. It consists of finding the face and the hat of a famous Walt Disney character, Uncle Scrooge, in a set of cartoon covers taken from two different Italian publications. The full dataset comprises 319 images, all of which show the character in various sizes and poses. Some pictures have multiple copies of the face, with or without the hat, whereas other images also show the faces of other Walt Disney characters, that are quite difficult to distinguish from Uncle Scrooge, f.i. Donald Duck, or other ducks. A training set, a validation set and a test set have been built randomly using, respectively, 127, 64 and 125 images from the dataset.

Images were encoded into Region Adjacency Graphs (RAGs). RAGs [17] are a classical image representation, where nodes stand for homogeneous regions and edges encode the relationship “is adjacent to” (Fig. 1). Since adjacency is a symmetric relationship, the obtained graphs are undirected. Previous attempts to elaborate RAGs with Recursive Neural Networks [4, 3], showed encouraging results, even if RAGs had to be transformed into acyclic graphs, because of the limitations of Recursive Neural Networks. On the contrary, GNNs are able to directly process undirected graphs, so no preprocessing was needed.

The RAGs have labels on both nodes and edges: node labels contain features of the corresponding image regions, while edge labels store information on adjacent regions pairs⁴ (Fig. 3). Tab. 1 lists all the features in detail. Each node n also has a manually assigned tar-

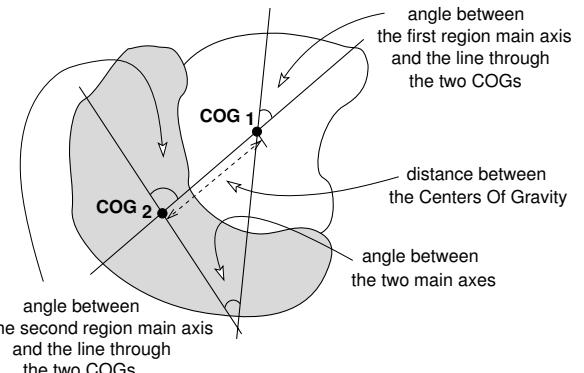


Figure 3. The geometric features stored in each edge label. COG is an abbreviation for “center of gravity”.

get t_n which is positive for regions belonging to Uncle Scrooge’s face or hat and negative elsewhere. The targets were assigned using a graphical interface that shows the segmented image and allows to select the regions that compose each object.

In order to obtain the homogeneous regions a segmentation procedure composed by three steps was adopted. In the first, each image was filtered using the Mean Shift algorithm [5], that acts like a low-pass filter in a mixed 5D feature space composed by the tristimulus

⁴ The used features have been computed using the LTI-Lib library [15], developed by the Chair of Technical Computer Science, RWTH-Aachen University, and released under the GNU Lesser General Public License.

Table 1. Features contained in node and edge labels

Label type	Feat. type	Description
Node label	color	average color
		area
		perimeter
	geometric	coordinates of the center of gravity (COG)
		coordinates of the bounding box
		$compactness: 4 * \pi * area / (perim.)^2$
		$eccentricity$
		main axis orientation
		inertia parallel. and orthog. to main axis
	moments	Several central moments m_{xy} of various orders w.r.t. x and y : $m_{02}, m_{03}, m_{11}, m_{12}, m_{20}, m_{21}, m_{30}$
		RGB-color distance between the average colors of the two regions
Edge label	geometric	distance between the two COGs
		angle between the main axes
		angle between the main axis of the first region and the line through the COGs
		angle between the main axis of the second region and the line through the two COGs

values in CIE-Luv color space and the bidimensional image coordinates (x, y). The main advantage of this technique is that it is especially designed to preserve gradient discontinuities. Thus smallest textures, often oversegmented by techniques employing only color information, are heavily smoothed, while significant edges remain almost crisp, easing region boundaries detection. This filtering step plays a key role in removing the noise introduced by image digitalization and JPEG compression⁵.

Then, a k-means [12] color quantization procedure was adopted to locate an initial large number of homogeneously colored regions. Finally, pairs of adjacent regions were recursively merged, using a heuristic function that evaluated similarities in order to choose the best pair. Such a procedure was repeated until the desired final number of regions was achieved. The optimal number of regions was automatically defined using a function that estimates the image complexity through the average value of the luminance gradient.

Several tests were performed using GNNs with three different architectures. More precisely, the state dimensions s and the number of hidden units r in the neural networks that implement the functions f_w and g_w were varied⁶. In the first GNN architecture, $s = 3$ and $r = 3$ hold, in the second architecture $s = 5$ and $r = 5$, and in third architecture $s = 10$ and $r = 10$.

To evaluate the importance of the subsymbolic information and the effectiveness of the structural representation of objects, the GNN model was compared with a three layered feedforward neural networks (MLP), that solved the problem using only the node labels as input patterns. Since the MLPs cannot exploit the graph connectivity, while GNNs can, the performance difference is mostly due to the subsymbolic information embedded in RAG topology.

Table 2 shows the accuracies obtained by the various architectures measured by the percentage of the correctly classified regions.

⁵ Mean Shift algorithm was implemented using the LTI-Lib library [15].

⁶ For sake of simplicity, in all the experiments the two neural network were three layered feedforward networks with the same number r of hidden neurons.

Table 2. Percent of correctly predicted regions in the positive examples (Object), in the negative (Non-obj.), and on average (Total)

Model	Configuration	Object	Non-obj.	Total
MLP	10 hidden u.	84.39%	68.91%	70.44%
	20 hidden u.	85.15%	69.45%	70.90%
	30 hidden u.	84.96%	69.27%	70.73%
GNN	state dimension = 3 3 hidden u. in funct. f 3 hidden u. in funct. g	78.90%	75.97%	74.72%
	state dimension = 5 5 hidden u. in funct. f 5 hidden u. in funct. g	77.29%	78.00%	77.93%
	state dimension = 10 10 hidden u. in funct. f 10 hidden u. in funct. g	78.81%	78.54%	78.56%

MLPs reached very high accuracies on the class of positive examples and quite low accuracies on the other nodes, while GNNs behaved similarly on the two classes. The reason is that the positive regions have features that belong to a restricted set, while the features of the negative regions have a wider range. Thus, false positives are more frequent than false negatives in MLPs. On the other hands, GNN are able to exploit also the information given the regions adjacent to the possible face, that usually represents the clothes or the hat. Such an information allows to largely decrease the number of false positives. Notice that the unbalanced results obtained by MLPs has a negative impact on the total accuracy of the MLPs w.r.t. the results achieved by GNNs, as clearly shown in Tab. 2. The error function considers both the positive and the negative examples, thus the training tried to force the MLPs to increase the performance on the negative examples, eventually decreasing the accuracy on the positive examples, but such a goal cannot be reached because of the task difficulty.

Since the dimension of the regions greatly varies in the dataset and larger regions are more important than smaller regions in object localization applications, the obtained networks were also evaluated on the base of the number of pixels that have been correctly classified. Table 3 shows the achieved results with accuracy defined as the ratio between the correctly predicted pixels and the total number of pixels of the image.

Table 3. Percent of correctly predicted pixels in the positive examples (Object), in the negative (Non-obj.), and on average (Total)

Model	Configuration	Object	Non-obj.	Total
MLP	10 hidden u.	83.72%	82.68%	82.73%
	20 hidden u.	86.76%	84.07%	84.20%
	30 hidden u.	83.92%	84.55%	84.52%
GNN	state dimension = 3 3 hidden u. in funct. f 3 hidden u. in funct. g	76.49%	87.91%	87.38%
	state dimension = 5 5 hidden u. in funct. f 5 hidden u. in funct. g	75.70%	88.92%	88.30%
	state dimension = 10 10 hidden u. in funct. f 10 hidden u. in funct. g	73.91%	88.56%	87.88%

Comparing the results of Tab. 2 and Tab. 3, we notice that all neural models better classify the largest regions w.r.t. the smallest ones. In fact, it can be shown that the mean size of the mistaken regions is about a half of the mean size of the correctly predicted ones, in all tests and with all architectures. This behavior is probably due to the fact that geometric features are likely to be progressively less significant when the regions become too small.

Finally, the correctly processed images were counted. In this case, an image was defined as correctly processed if the number of correctly classified pixels exceeded a given threshold. Table 4 shows the achieved accuracies using a 80% threshold and a 90% threshold. The

Table 4. Percent of images where the total pixel-based accuracy exceeded 80% and 90%

Model	Configuration	Acc. \geq 80%	Acc. \geq 90%
MLP	10 hidden u.	75.59%	17.32%
	20 hidden u.	76.38%	19.69%
	30 hidden u.	78.74%	20.47%
GNN	state dimension = 3 3 hidden u. in funct. f 3 hidden u. in funct. g	83.46%	41.73%
	state dimension = 5 5 hidden u. in funct. f 5 hidden u. in funct. g	90.55%	51.18%
	state dimension = 10 10 hidden u. in funct. f 10 hidden u. in funct. g	82.68%	52.76%

results show that GNNs outperform MLPs because the structural information make GNNs less prone to spurious features. In fact, sets of adjacent positive regions reinforce themselves mutually, and the same happens for each cluster of negative regions.

Finally it is worthy to remark that GNNs succeeded in correctly classifying over the ninety percent of the pixels for more than a half of the images. Since the faced problem was particularly difficult, those results are encouraging and prove that the application of the GNN model on structural domains is viable.

4 Conclusions

This paper has presented the preliminary results obtained by the application of the Graph Neural Network model to an object localization task. The results were encouraging and have shown that the model can really combine the symbolic and the subsymbolic information stored in structured data. Future matter of research includes a more systematic experimentation of the GNN model on object localization problems and the application of the model to other tasks where the information is naturally represented by graphs as relational learning problems and social network applications.

REFERENCES

- [1] L.B. Almeida, ‘A learning rule for asynchronous perceptrons with feedback in a combinatorial environment’, in *Proceedings of the IEEE International Conference on Neural Networks*, eds., M. Caudill and C. Butler, volume 2, pp. 609–618, San Diego, 1987, (1987). IEEE, New York.
- [2] P. Baldi and G. Pollastri, ‘The principled design of large-scale recursive neural network architectures-dag-rnns and the protein structure prediction problem’, *Journal of Machine Learning Research*, **4**, 575–602, (2003).
- [3] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli, ‘Recursive neural networks for processing graphs with labelled edges: Theory and applications’, *Neural Networks - Special Issue on Neural Networks and Kernel Methods for Structured Domains*, **18**, 1040–1050, (October 2005).
- [4] Monica Bianchini, Paolo Mazzoni, Lorenzo Sarti, and Franco Scarselli, ‘Face spotting in color images using recursive neural networks’, in *Proceedings of the 1st ANNPR Workshop*, Florence (Italy), (2003).
- [5] Dorin Comaniciu and Peter Meer, ‘Mean shift: A robust approach toward feature space analysis’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(5), 603–619, (2002).
- [6] F. Costa, P. Frasconi, V. Lombardo, and G. Soda, ‘Towards incremental parsing of natural language using recursive neural networks’, *Applied Intelligence*, **19**(1–2), 9, (July 2003).
- [7] E. Francesconi, P. Frasconi, M. Gori, S. Marinai, J.Q. Sheng, G. Soda, and A. Sperduti, ‘Logo recognition by recursive neural networks’, in *GREC ’97: Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems*, eds., Karl Tombre and Atul K. Chhabra, 104–117, Springer-Verlag, (1998).
- [8] P. Frasconi, M. Gori, and A. Sperduti, ‘A general framework for adaptive processing of data structures’, *IEEE Transactions on Neural Networks*, **9**(5), 768–786, (1998).
- [9] M. Gori, G. Monfardini, and F. Scarselli, ‘A new model for learning in graph domains’, in *Proc. International Joint Conference on Neural Networks (IJCNN2005)*, pp. 729–734, (2005).
- [10] M. Hagenbuchner, A. Sperduti, and Ah Chung Tsoi, ‘A self-organizing map for adaptive processing of structured data’, *IEEE Transactions on Neural Networks*, **14**(3), 491–505, (May 2003).
- [11] B. Hammer and J. Jain, ‘Neural methods for non-standard data’, in *Proceedings of the 12th European Symposium on Artificial Neural Networks*, ed., M. Verleysen, pp. 281–292, (2004).
- [12] J. A. Hartigan and M. A. Wong, ‘A k-means clustering algorithm’, *Applied Statistics*, **28**, 100–108, (1979).
- [13] Mohamed A. Khamsi, *An Introduction to Metric Spaces and Fixed Point Theory*, John Wiley & Sons Inc, 2001.
- [14] E. Krahmer, S. Erk, and A. Verleg, ‘Graph-based generation of referring expressions’, *Computational Linguistics*, **29**(1), 53–72, (2003).
- [15] LTI-Lib. An object oriented library with algorithms and data structures for image processing and computer vision. Developed at the Chair of Technical Computer Science (Lehrstuhl fuer Technische Informatik) LTI at the Aachen University of Technology. Available at <http://ltlib.sourceforge.net/>.
- [16] M. E. J. Newman, ‘From the cover: The structure of scientific collaboration networks’, *Proc. National Academy of Sciences*, 404–409, (2001).
- [17] T. Pavlidis, *Structural pattern recognition*, Springer, Series in Electrophysics, 1977.
- [18] F.J. Pineda, ‘Generalization of back-propagation to recurrent neural networks’, *Physical Review Letters*, **59**, 2229–2232, (1987).
- [19] M. Riedmiller and H. Braun, ‘A direct adaptive method for faster back-propagation learning: the rprop algorithm’, in *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pp. 586–591, San Francisco, CA, USA, (1993).
- [20] F. Scarselli, M. Gori, G. Monfardini, Ah Chung Tsoi, and M. Hagenbuchner, ‘A new neural network model for graph processing’, Technical Report DII 01/05, Department of Information Engineering, University of Siena, (2005).
- [21] A. Sperduti and A. Starita, ‘Supervised neural networks for the classification of structures’, *IEEE Transactions on Neural Networks*, **8**, 429–459, (1997).
- [22] H.C. van Assen, M. Egmont-Petersen, and J.H.C. Reiber, ‘Accurate object localization in gray level images using the center of gravity measure: accuracy versus precision’, *IEEE Transactions on Image Processing*, **11**(12), 1379–1384, (December 2002).
- [23] M.-H. Yang, J. Kriegman, and N. Ahuja, ‘Detecting faces in images: A survey’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(1), 34–58, (January 2002).

This page intentionally left blank

10. Robotics

This page intentionally left blank

Situation Assessment for Sensor-Based Recovery Planning

Abdelbaki Bouguerra and Lars Karlsson and Alessandro Saffiotti¹

Abstract. We present an approach for recovery from perceptual failures, or more precisely anchoring failures. Anchoring is the problem of connecting symbols representing objects to sensor data corresponding to the same objects. The approach is based on using planning, but our focus is not on the plan generation per se. We focus on the very important aspect of situation assessment and how it is carried out for recovering from anchoring failures. The proposed approach uses background knowledge to create hypotheses about world states and handles uncertainty in terms of probabilistic belief states. This work is relevant both from the perspective of developing the anchoring framework, and as a study in plan-based recovery from epistemic failures in mobile robots. Experiments on a mobile robot are shown to validate the applicability of the proposed approach.

1 Introduction

Mobile robots and other autonomous systems acting in the real world face the problem of uncertainty, and uncertainty can cause even the best laid plans to fail. Therefore, the ability to recover from failures is vital for such systems. AI planning is one of the tools used to recover from failures. Recovery is viewed as a planning task where the initial state of the planner is the unexpected state encountered by the agent, and the goal is to reach a nominal state where the plan can be continued. Most robotic systems employing planning [9, 1] focus largely on generation and repair of plans. However, having a good planner is not sufficient, as a planner fed with the wrong initial state will produce an inadequate plan. Consequently, a functionality for situation assessment is also needed. By situation assessment we mean the process of identifying what went wrong, and the creation of a description that reflects what is known about the real world state.

In this paper, we study the problem of situation assessment for recovering from anchoring failures in mobile robotics using sensor-based planning [3]. Anchoring is the problem of establishing the correspondence between a symbol and a percept. A type of anchoring failure is when one cannot determine which percept corresponds to a given symbol due to ambiguity: there are several plausible percepts. This requires the robot to generate a description of the current situation and generate a plan that disambiguates that situation. However, the approach in [3] does not quantify uncertainty, so there is no way to distinguish between more likely and less likely candidates. All candidates are considered equally important.

The main contribution of this paper is an approach for automatic situation assessment for recovery planning in terms of probabilistic belief states. The recovery module can assign probabilities to what

perceived object should be anchored to the given symbol, and it can generate plans that disambiguate the current situation with a threshold probability instead of plans that always have to succeed. Finally, it can choose to anchor an object to a symbol with a degree of certainty; absolute certainty is not required. In fact there are situations where certain relevant properties of an object cannot be determined, and absolute certainty about which is the correct object is not attainable. Besides, the robot might have only a limited amount of time for planning. The approach in [3] did not permit this.

Besides probabilities, there are two other contributions relative to [3]. The first one is the use of background knowledge to fill in missing information about perceived objects in terms of conditional probabilities. The second is the handling of new candidate objects that are perceived at run-time. We do so by making a closed world assumption during plan generation (by considering only those relevant objects perceived so far), and we use a monitor process to check that the closed world assumption is not violated. When it is violated, a new plan that takes the new objects into account is generated.

This work is relevant both from the perspective of developing the anchoring framework, and as a case study in the more general problem of sensor-based recovery.

This paper is organized as follows: in the next section we review the concept of perceptual anchoring and the problem of ambiguity. In section 3, we present our method for automatically recovering from ambiguous anchoring situations; this is the central part of the paper. Section 4 presents 3 different experiments intended to test our approach, and section 5 provides some discussion.

2 Overview of Perceptual Anchoring

Anchoring is the process of establishing and maintaining the correspondence between the perceptual data and symbolic abstract representation that refer to the same physical object [4]. Intelligent embedded systems using symbolic representations, such as mobile robots, need to perform some form of anchoring in order to achieve their tasks. Consider a mobile robot, equipped with a vision system and a symbolic AI planner, trying to find dangerous gas bottles in a building at fire. In order to execute the ‘GoNear (bottle1)’ action, where the symbol ‘bottle1’ refers to an object described as ‘a green gas bottle’, the robot must make sure that the vision percept it is approaching is the one of the object whose identity is ‘bottle1’.

The symbolic description of an object with symbol o is denoted $Desc(o)$. A percept π consists of information about an object derived from sensor data (e.g. a video image), such as estimates of shape, color and other properties. These estimates are often uncertain; for instance, the vision module we use assigns degrees to the values of properties and to the classification of objects.

¹ Center for Applied Autonomous Sensor Systems - AASS;
Örebro University; Sweden; Web: <http://www.aass.oru.se>; Email: {abdelbaki.bouguerra,lars.karlsson}@tech.oru.se; asaffio@aass.oru.se

In order to decide which percept π to anchor to a given symbol o , one needs to match $Desc(o)$ against the properties of π . The result is either *no* match, a *partial* or a *complete* match [5]. A percept π is said to be completely matching o if all the properties of $Desc(o)$ match those of π , and partially matching if it at least a property of the description $Desc(o)$ cannot be determined to match its counterpart in π due to uncertainty.

Another important distinction is whether a symbol o is making a definite reference, to exactly one object in the world ("the red ball"), or an indefinite reference, to any matching object ("a red ball").

2.1 Relational properties

There are situations when it is relevant to describe objects not just in terms of their properties but also their relations to other objects. An example is "the green garbage can that is near the red ball and the blue box". We consider the object that needs to be anchored, in the example "the green can", as the *primary object* and the other objects related to it, in the example "the red ball" and "the blue box", as *secondary objects* [3]. In this paper we use in particular binary relations and we allow for descriptions with several nested relations.

Definition 1 Let O denote the set of object symbols. A relational description of an object $o \in O$ having m binary relations ($R_{k;1 \leq k \leq m}$) with m secondary objects ($o_k;1 \leq k \leq m$) is denoted $Rel_{desc}(o)$ and it is defined recursively as:

$$Rel_{desc}(o) =_{def} Desc(o) \bigcup_{1 \leq k \leq m} \{R_k(o, o_k)\} \cup Rel_{desc}(o_k)$$

Anchoring a symbol with a relational description, involves considering the secondary objects too. They become new objects to anchor. In practice, we always limit the depth of the relational description.

Example: In a world configuration there are three objects $O = \{o_1, o_2, o_3\}$ such that o_1 is a red ball, o_2 is a blue box, and o_3 is a cup whose color is green. We have also observed that o_1 is *near* o_2 , and o_3 is *on* o_2 . The relational description of o_1 is given by:

$$\begin{aligned} Rel_{desc}(o_1) = & \{(shape\ o_1 = ball), (color\ o_1 = red)\} \cup \\ & \{(near\ o_1\ o_2), (shape\ o_2 = box), (color\ o_2 = blue)\} \cup \\ & \{(on\ o_3\ o_2), (shape\ o_3 = cup), (color\ o_3 = green)\} \end{aligned}$$

2.2 Anchoring with Ambiguities

There are situations where the anchoring module cannot create or maintain an anchor for a specific symbol from its percepts because of the presence of ambiguity, i.e. it cannot determine what anchoring candidate to choose. In order to detect and resolve ambiguities, we need to consider what candidates there are to anchor the given symbol, as well as the symbols of related objects.

Definition 2 A relational anchoring candidate for an object o having m binary relations ($R_{k;1 \leq k \leq m}$) with m secondary objects ($o_k;1 \leq k \leq m$) is represented by a list $(\pi_0, (\pi_{1,1} \dots), (\pi_{2,1} \dots), \dots, (\pi_{m,1} \dots))$ such that π_0 is a candidate percept for the primary object o , and for each secondary object o_k , a (possibly empty) list $(\pi_{k,1} \dots)$ of all candidate percepts satisfying $Rel_{desc}(o_k)$ and relation $R_k(o, o_k)$.

Notice that the same definition applies recursively to relational anchoring candidates for secondary objects. In fact, a relational anchoring candidate can be easily represented using an *and-or* tree where

the *and* nodes represent the relations and the *or* nodes represent the candidate percepts satisfying the relation of the parent node.

A relational anchoring candidate is *completely matching* if π_0 completely matches $Desc(o)$ (the primary object), and for each secondary object o_k there is only one (definite case) or at least one (indefinite case) candidate percept $\pi_{k,j}$ completely matching $Desc(o_k)$. The definite/indefinite distinction here refers to whether the secondary object symbol is definite/indefinite.

A relational anchoring candidate is *partially matching* if for some o_k there is no completely matching percept $\pi_{k,l}$. A special case is when there is no $\pi_{k,l}$ at all, i.e. the candidate list for o_k is empty.

Once each relational candidate for the desired object (primary) has been classified this way, the entire situation is handled. The anchoring module detects the presence of ambiguity on the basis of the number of complete and partial anchoring candidates, and whether the description is definite or indefinite [3]. In short, the definite case is ambiguous unless there is exactly one candidate, and it is completely matching (more than one complete match is considered a conflict). The indefinite case is ambiguous if there is no completely matching candidate. In case of ambiguity the recovery module is invoked to find a plan to acquire additional information about primary and/or secondary candidate objects if necessary.

Example The robot is presented with the symbolic description " $g1$ is a garbage can near a red ball with a mark" and given the task to go near $g1$. To do this, the robot needs to anchor the symbol $g1$. Consider a situation where a garbage can π_0 and a red ball $\pi_{1,1}$ are perceived, but no mark is visible. We have a situation where we have a singleton set $\{(\pi_0, (\pi_{1,1}))\}$ of relational candidates. There is one fully matching percept for the primary object, and one partial match for the secondary object: the mark was not observed. Consequently the entire relational candidate is a partial match, giving rise to ambiguity. Thus, to be sure that the observed garbage can is the requested one, the red ball has to be checked for marks from other viewpoints.

3 Recovering from Anchoring Failures

A recovery situation in anchoring typically occurs when the robot is executing some higher-level plan and encounters an ambiguous situation. Such a situation is handled in six steps:

1. Detection: the problematic situation is detected, and the top-level plan is halted.
2. Situation Assessment: the recovery module analyzes the problematic situation. If the situation is found to be recoverable, i.e. due to uncertain perceptual information (see [3] for more details), it formulates a belief state consisting of a set of possible worlds with probabilities. This step is achieved by considering the properties of the requested object and of the perceived objects to generate different hypotheses for which of the perceived objects corresponds to the requested one.
3. Planning: the planner is called to achieve the goal of disambiguating the situation and anchoring the requested object.
4. Plan Execution: the plan is executed, and either the requested object is found and identified and can be anchored, or it is established that it cannot be identified.
5. Monitoring: if during the execution of the recovery plan, new perceived objects are encountered that completely or partially match the primary or a secondary object, then go back to step 2.
6. Resume: if recovery was successful, the top-level plan is resumed.

The authors in [3] describe how the recovery module builds the initial belief state, that encodes the different hypotheses of anchoring

the desired object to one of the perceived candidates. In summary, it constructs one hypothesis, for each relational anchoring candidate, consisting of a positive (completely matching) description of the chosen candidates, and negative (non-matching) descriptions of the remaining candidates. Hypotheses with zero or two or more candidates may also be generated. These different hypotheses are then combined in a disjunction to generate a (crisp) belief state.

When we introduce probabilities, we need to be able to handle dependencies between different candidates in a graceful manner, in particular when the same percept appears more than once. Consequently, we have taken a reversed approach: instead of starting from the different candidates, we simply start from what is known about the different perceived objects, and generate a belief state from that. While doing this, we can also use background knowledge to fill in missing information about percepts. Next, we go through the possible worlds in the belief state and classify them according to what relational anchoring candidate they agree with (if any).

3.1 Background knowledge

Ambiguous situations typically involve partial matches of some perceived objects. This occurs when a certain property or relation is observed with uncertainty, or neither it nor its opposite was observed at all. Situation assessment involves considering how probable it is that those properties hold. There are two sources from which this information can be obtained.

First, the vision system (and other sensing systems) can provide us with matching degrees, which can serve as weights when assigning a probability distribution to a property or relation. For instance, if it could not be determined whether a perceived object π_1 was red or orange but red is matching better than orange, we might assert in our planner language (color pi-1 = (red 0.6) (orange 0.4)), which means $P(\text{color}(\pi_1) = \text{red}) = 0.6$ and $P(\text{color}(\pi_1) = \text{orange}) = 0.4$.

Second, we can use background knowledge expressed as conditional probabilities. This background knowledge is encoded as probabilistic assertion rules that can specify conditional and prior probability distributions over the values of uncertain properties of perceived objects. Probabilistic assertion rules are a part of a language used for the forward-searching planners [8, 2] to describe the results of the application of an action in a belief state. These action results may involve conditional probabilistic effects, and hence the same representation can also be used to encode background knowledge.

Example: The following rule describes the conditional probability of a perceived object $?o$ containing (has) a substance (milk, tea or nothing) given its shape (cup, bowl, or something else).

```
(forall (?o) (perc ?o)
  (cond
    ((shape ?o = cup)
     (has ?o = (milk 0.4) (Tea 0.4) (nothing 0.2)))
    ((shape ?o = bowl)
     (has ?o = (milk 0.2) (Tea 0.3) (nothing 0.5)))
    ((true) (has ?o = (Tea 0.1) (nothing 0.9))))
```

This rule can be used when we see an object but do not know what it contains. For instance, percepts that have the shape of a cup are believed to contain either milk, or tea (with 0.4 probability), or nothing (with 0.2 probability). Notice how conditional probabilities are defined using the `cond` form which specifies conditional outcomes. It works like a LISP `cond` where each clause consists of a test (which may refer to uncertain properties) followed by a consequent formula.

The `forall` assertion formula allows to iterate over elements of a certain type (here `perc` for percept), to execute an assertion formula (here the `cond` form).

Assertion formulas are applied to belief states, and belief states are probability distributions over sets of possible worlds (crisp states). The assertion formula is applied to each possible world in the belief state, resulting in new possible worlds. These new possible worlds and their probabilities constitute the new belief state(s). Note that the belief state explicitly represents a joint probability distribution over the uncertain properties and relations in the situation.

3.2 Situation Assessment

Once the recovery module is notified of the failure to anchor an object o due to ambiguity, it starts a situation assessment process aiming at creating a belief state representing the different states the world may actually be in, given present uncertainties. The situation assessment process takes a set of relational anchoring candidates as input, and is performed in four steps.

1) Initialization: a number of properties not related to the anchoring candidates are assessed, such as the position of the robot and the topology of the room.

2) Description generation: the descriptions $Desc(\pi_j)$ and $R_k(\pi_j, \pi_k)$ are computed for all the perceived objects π_j in the relational anchoring candidates. The properties and relations that are considered are those appearing in $Rel_{desc}(o)$ for the object o such that π_j is a candidate object for o . Uncertain properties are estimated in the manner described in section 3.1, by using matching degrees and background knowledge. The result of the first step is a belief state representing a joint probability distribution over all uncertain properties and relations of the perceived objects in the candidates.

3) Classification: the possible worlds of the belief state are partitioned into three different sets for the definite case and two sets for the indefinite case. For the definite case, one partition contains the possible worlds where there is a unique matching candidate. A second partition contains those worlds where there is a conflict due to the presence of more than one matching candidate, and a third set includes worlds where there is no matching candidate. Partitioning relies on the evaluation of three existential formulas derived from $Rel_{desc}(o)$. Those formulas test if there is exactly one, two or more, and no relational anchoring candidate that matches $Rel_{desc}(o)$, respectively. In these formulas, the object names in $Rel_{desc}(o)$ are replaced by existentially quantified variables.

When the recovery module deals with an indefinite case, the partitioning yields only the set where there is no matching candidate and the set where there is at least a matching candidate (in the indefinite case there is no situation of conflict).

Sometimes, one might want to provide more weight to the possible worlds where there is exactly one anchoring candidate that matches. After all, if the robot was ordered to go to "the container with milk" it might be reasonable to consider it likely that there is exactly one such object. Hence, at this step one may discount the probability of the possible worlds with none or too many matching candidates using a discount factor α and then renormalize the probability distribution of the possible worlds to sum to one.

4) Labeling: The formula $(\text{anchor } o \ \pi)$ is added to each possible world where percept π is a fully matching candidate, and the formula $(\text{anchor } o \ \text{null})$ to the set of non-matching worlds (and conflict worlds for the definite case) to ascertain that the desired object o is not possible to anchor. The percepts π are those binding the variables in the existential formulas in step 3.

Example Assume that the plan executor could not anchor object $C1$ described as the container with milk which is near the fridge, because there are two perceived container objects π_c and π_b near the fridge π_f : π_c is a green cup and π_b is a blue bowl. Let's also assume that the recovery module uses the probabilistic conditional rule from section 3.1.

In step one, it is asserted that the robot is at the entrance of the coffee room and so on.

There are two relational candidates: $(\pi_c(\pi_f))$ and $(\pi_b(\pi_f))$. Step 2 consists of computing the description of all the perceived objects appearing in the relational candidates and relations among them to build an initial belief state for the subsequent steps.

The descriptions we obtain are a set Rel of two relations $\{(near \pi_c \pi_f), (near \pi_b \pi_f)\}$, and the following descriptions of the different perceived objects:

$$\begin{aligned} Desc(\pi_c) &= \{(perc \pi_c), (shape \pi_c = cup), (color \pi_c = green)\} \\ Desc(\pi_b) &= \{(perc \pi_b), (shape \pi_b = bowl), (color \pi_b = red)\} \\ Desc(\pi_f) &= \{(perc \pi_f), (shape \pi_f = fridge)\} \end{aligned}$$

We assert these descriptions, apply the background knowledge in section 3.1 and obtain a belief state bs with four possible worlds w_1, \dots, w_4 such that:

$$\begin{aligned} w_1 &= \{(has \pi_c = milk), (has \pi_b = (Tea 3/8)(nothing 5/8))\} \\ w_2 &= \{(has \pi_c = milk), (has \pi_b = milk)\} \\ w_3 &= \{(has \pi_c = (Tea 4/6)(nothing 2/6)), \\ &\quad (has \pi_b = (Tea 3/8)(nothing 5/8))\} \\ w_4 &= \{(has \pi_c = (Tea 4/6)(nothing 2/6)), (has \pi_b = milk)\} \end{aligned}$$

In addition, they all contain $Rel \cup Desc(\pi_c) \cup Desc(\pi_b) \cup Desc(\pi_f)$. The probability distribution over the possible worlds is:

$$\begin{aligned} p(w_1) &= 0.4 \cdot 0.8 = 0.32; p(w_2) = 0.4 \cdot 0.2 = 0.08 \\ p(w_3) &= 0.6 \cdot 0.8 = 0.48; p(w_4) = 0.6 \cdot 0.2 = 0.12 \end{aligned}$$

We compute $p(w_1)$ as the joint probability of π_c containing milk and π_b not containing milk, and so on.

In step 3, the situation assessment module classifies the worlds according to the number of matching candidates. In worlds w_1 and w_4 there is one and only one matching candidate. In world w_2 , there are two matching candidates, therefore we have a conflict. Finally in world w_3 there is no candidate object matching $Rel_{desc}(c1)$ i.e. the test formula for no matching candidates, $\neg \exists x_1, x_2 ((has x_1 = milk) \wedge (near x_1 x_2) \wedge (shape x_2 = fridge))$, holds in w_3 .

In step 4, the situation assessment module adds $(anchor c_1 \pi_c)$ to w_1 , $(anchor c_1 \pi_b)$ to w_4 , $(anchor c_1 \text{null})$ to both w_2 and w_3 .

3.3 Planning and Monitoring

3.3.1 Planning and Execution

Once the recovery module has created the belief state encoding the possible worlds, it passes it to the planner together with the goal of finding an anchor to the requested object. We use probabilistic conditional planners for stochastic partially observable domains with actions and sensing modeled in a POMDP-equivalent manner [8, 2].

The generated plans incorporate motion actions and sensing actions aiming at collecting more information about the unknown properties of the perceived objects. They are guaranteed to solve the planning problem (if it is solvable) with a certain probability of success.

The anchoring plan is then sent to the plan execution and monitoring module. Fig. 1 shows the global architecture of our mobile robot. The recovery module is part of the block labeled Executor/Monitor.

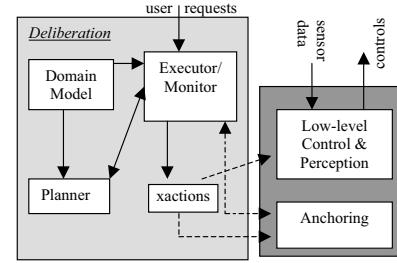


Figure 1. Plan execution and monitoring architecture

The actions such as $(smell \pi_c)$ and $(move-to pos2)$ are translated into executable perceptual and movement tasks ($xactions$ in the figure) that are sent to the low-level control architecture for execution. The anchoring module provides the plan executor/monitor with the results of perceptual task, which allow the latter to keep an updated belief state and to monitor how the plan progresses. The domain model module in the figure stores planning domains (actions, axioms and predicates) as well as the different conditional rules for building the background knowledge.

3.3.2 Closed World Assumption Monitoring

The recovery plans are generated under the assumption that all relevant candidates have been observed, with the exception of the explicit “missing objects” used when there is no candidate at all. However, there may be additional candidate objects that are not visible from the initial position of the robot, but become visible as the robot moves around while executing its recovery plan. The anchoring system regularly checks for the appearance of new percepts matching the description of the requested object. If such a percept is detected, the assumption of knowing all relevant objects has been violated, and the current recovery plan is considered obsolete. Hence, a new initial belief state is produced, taking into account the previous perceived objects and the information we have gained of those, as well as the new perceived object, and a new recovery plan is generated. This new plan replaces the old one.

Replanning for new candidate objects is not only used for the unexpected discovery of a new object: it is also instrumental in the case where the robot was explicitly searching for a candidate, and found a partially matching one. In such a situation, one cannot jump to the conclusion that the correct object has been found, but must plan to find more information about the object in question (together with the other remaining candidate objects).

4 Experiments

In [3] it has been experimentally demonstrated that the ability to recover from anchoring failures can improve the robustness of the behaviors of a mobile robot. Here, we intend to demonstrate how our approach can handle three different scenarios where probabilities are required. Our experiments are intended to show the feasibility of our approach; not to evaluate its performance. The experiments were run on a Magellan Pro Research Robot equipped with a CCD camera. The recovery planning time was always less than two seconds.

Scenario 1 serves to demonstrate that we can handle cases where there is no recovery plan that is certain to succeed. Our robot is to approach a marked gas bottle (indefinite reference). From its initial position, it

perceives two gas bottles π_1 and π_2 , but no mark is perceived on them. The mark may be on one of four sides of each gas bottle. However, the presence of obstacles prevents observing them from all sides. The situation assessment produced a belief state consisting of four possible worlds w_1, \dots, w_4 with a uniform probability i.e. $p(w_i) = 1/4; 1 \leq i \leq 4$. Each world reflects if π_1 or π_2 is having a mark, and if it had a mark, on which side it is. A plan was generated in which the robot is to move to the different accessible positions to look for a mark on the perceived gas bottles. When observation actions are performed, the probabilities of the possible worlds are updated accordingly. For instance, if no mark is seen on one side of π_1 , the probability that π_1 is a match decreases. In one of the runs, the robot had not found a mark after observing π_1 from three sides, and π_2 from two sides. At this stage, the anchoring module decided to anchor the more likely candidate π_2 but with a low degree of certainty. The scenario was also run successfully under different configurations (in terms of the locations of the gas bottles, whether they were marked, and observability of the marks on them).

Scenario 2 concerns run-time handling of new objects. Again, our robot is to approach a gas bottle with a mark upon it (indefinite reference). This time, only one gas bottle π_1 is initially perceived. The situation assessment produced a belief state with two uniformly distributed possible worlds w_1 (π_1 is marked), and w_2 (π_1 is not marked). In this scenario, all sides are observable, therefore in w_1 there is a probability distribution for which side, of the gas bottle, has the mark. When the recovery plan is executed, and the robot has moved to a different position, it detects a second gas bottle initially occluded by the first one. The situation is reassessed (to include the new perceived gas bottle) and a new recovery plan is produced that includes actions to check both the old and the new gas bottle. Also, this scenario was successfully run under different configurations (which of the bottles was marked, and which side of the bottle the mark was on).

Scenario 3 involves using background knowledge to resolve an ambiguous situation, where planning time is very short. The robot has to achieve the same task as before, but in addition there is the background knowledge that 90% of brown gas bottles have a mark, and only 20% of green ones have a mark. From its initial position, the robot perceives two gas bottles: one green and one brown. Using the background knowledge, the situation assessment procedure notices that the brown gas bottle has a higher chance of matching the description and decides to anchor it directly.

5 Discussion and Conclusions

We have presented a probabilistic approach to recovering from ambiguous anchoring situations, i.e. when one cannot determine which percept corresponds to a given symbol. The only previous approach [3] was non-probabilistic, and could not distinguish between more likely and less likely candidates, nor could it handle cases where the ambiguous situation could only partially be resolved.

Our approach (just like the previous one) builds on the steps detection, situation assessment, plan generation and execution. In addition, we have added monitoring for the appearance of new relevant objects. We have focused on the assessment step, where we have introduced the possibility to use matching degrees for percepts as well as background knowledge specifying conditional probabilities to determine the probability distributions for uncertain or unknown properties and relations.

The background knowledge is specified in terms of assertion formulas in the language of our planners. That has the advantage that they can be applied directly to the belief states that are to be used by

the planners, but they do not offer a compact representation of uncertainty. Therefore, we are presently considering other representations for our situation assessment steps. Probabilistic relational models (PRMs) [6] generalize Bayesian Networks to represent first order knowledge i.e. knowledge involving classes of objects where some attributes in a specific class are uncertain and may depend on attributes of the same class or another one. Relational Bayesian Networks [7] are also a tool that uses Bayesian Networks to represent objects explicitly and relations among them. Finally, the Bayesian language BLOG [10] is a tool that can be used to create first order probabilistic models involving unknown number of objects. BLOG deals with possible worlds of different numbers of objects and uncertainty about their attributes. That might offer a way to reason and plan explicitly about unobserved but possibly present objects. However, we have favored handling that problem by interleaving planning and execution, a desirable characteristic for acting under uncertainty due to lack of information [11].

The particular problem we have addressed in this paper concerns the important problem of anchoring: if a mobile robot with symbolic cognitive layer is to interact with different objects in its environment, it will frequently need to anchor those objects. A robust anchoring mechanism is then a prerequisite for robust behaviors. But the problem is also interesting as a case of what can be called epistemic recovery: if the plans of a robot or other agent is stalled due to lack of information, how can that be handled? We believe our work has demonstrated the usefulness of probabilistic sensor-based planning, and the important role of situation assessment in such unexpected situations.

ACKNOWLEDGEMENTS

This work has been supported by the Swedish KK foundation and the Swedish research council.

REFERENCES

- [1] M. Beetz, ‘Runtime plan adaptation in structured reactive controllers’, in *Proc. of the 4th Int. Conf. on Autonomous Agents*, pp. 19–20, (2000).
- [2] A. Bouguerra and L. Karlsson, ‘Hierarchical task planning under uncertainty’, in *Proc. of the 3rd Italian Workshop on Planning and Scheduling*, (2004).
- [3] M. Broxvall, S. Coradeschi, L. Karlsson, and A. Saffiotti, ‘Recovery planning for ambiguous cases in perceptual anchoring’, in *Proc. of the 20th National Conf. on Artificial Intelligence*, pp. 1254–1260, (2005).
- [4] S. Coradeschi and A. Saffiotti, ‘Anchoring symbols to sensor data: preliminary report’, in *Proc. of the 17th National Conf. on Artificial Intelligence*, pp. 129–135, (2000).
- [5] S. Coradeschi and A. Saffiotti, ‘Perceptual anchoring of symbols for action’, in *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence*, pp. 407–412, (2001).
- [6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, ‘Learning probabilistic relational models’, in *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence*, pp. 1300–1309, (1999).
- [7] M. Jaeger, ‘Relational Bayesian networks’, in *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence*, pp. 266–273, (1997).
- [8] L. Karlsson, ‘Conditional progressive planning under uncertainty’, in *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence*, pp. 431–438, (2001).
- [9] S. Lemai and F. Ingrand, ‘Interleaving temporal planning and execution in robotics domains’, in *Proc. of the 19th National Conf. on Artificial Intelligence*, pp. 617–622, (2004).
- [10] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov, ‘Blog: Probabilistic models with unknown objects’, in *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence*, (2005).
- [11] I. Nourbakhsh, ‘Using abstraction to interleave planning and execution’, in *Proc. of 3rd Biannual World Automation Congress*, (1998).

Learning Behaviors Models for Robot Execution Control

Infantes Guillaume and Ingrand Félix and Ghallab Malik¹

Abstract. Robust execution of robotic tasks is a difficult problem. In many situations, these tasks involve complex behaviors combining different functionalities (e.g. perception, localization, motion planning and motion execution). These behaviors are often programmed with a strong focus on the robustness of the behavior itself, not on the definition of a “high level” model to be used by a task planner and an execution controller. We propose to learn behaviors models as structured stochastic processes: Dynamic Bayesian Network. Indeed, the DBN formalism allows us to learn and control behaviors with controllable parameters. We experimented our approach on a real robot, where we learned over a large number of runs the model of a complex navigation task using a modified version of Expectation Maximization for DBN. The resulting DBN is then used to control the robot navigation behavior and we show that for some given objectives (e.g. avoid failure, optimize speed), the learned DBN driven controller performs much better than the programmed controller. We also show a way to achieve efficient incremental learning of the DBN. We believe that the proposed approach remains generic and can be used to learn complex behaviors other than navigation and for other autonomous systems.

1 Introduction and Motivations

Tasks execution on autonomous robots is a very complex process: exploration rovers are expected to plan and properly execute their science observation missions; museum guide robots have to robustly execute their tour plans for visitors interested in a given subject; service robots must plan and execute daily activities for the elderly people they assist etc. The building blocks of these plans, i.e. the tasks and actions, can be quite complex. We refer to them as behaviors. These behaviors usually have an intrinsic complexity (e.g. navigation for a museum guide robot involve localization, perception, motion planning, etc.). Moreover, often no explicit model exists of how it performs in various environments. Last, even if a deterministic model of these behavior exists, it may not be appropriate to handle the intrinsic non-determinism of the environment and of the behavior execution outcomes. As a result, one has to cope with a planning problem where one must plan actions execution with poor model, or even, in some situation, with no model at all.

Some approaches [2] try to model actions deterministically, and apply a probabilistic bias afterward. This kind of approach may not be relevant for intrinsically stochastic systems, which are more common in real-world domain. Bayesian Networks [19] give to the authors of [9] the ability to model the actions and predict the effects of a manipulation task of a robot. But while this kind of model captures the stochastic nature of the system, it does not take into account its long-term dynamics.

Systems may be modeled as a Hidden Markov Model (HMM) [20]. In these models, the internal state is *hidden* to the observer, who

can only see the *observation*, that represent effects of the system on the environment. These stochastic models have proved quite adapted in domains such as speech recognition [20], modeling human activity [16], probabilistic plan recognition [5] and learning topological and metric maps [14]. In [10], the authors present an approach to models robot actions using HMM. Indeed, the resulting HMM actions can be used to recognize or to plan the modeled actions.

Yet, this representation does not allow the finer control of the action execution. An aspect of complex actions we want to address is that some of these activities may be controllable. Finding the proper parameter values with respect to the current environment and goal can be quite a challenge if again no model of the underlying action is available. Hence we would like to learn action models rich enough to allow us to use them to perform any of recognition, planning, control or supervision of the action. We propose to learn action models as highly structured stochastic processes: Dynamic Bayesian Network (DBN) [8].

We detail how such a model can be obtained from a number of real-world runs. We then show how it can be used to control the action itself while executing. We also sketch how this could be embedded in a more general controller able to supervise the action execution (to avoid failure) and to decide when the model has to be refined for new situations arising. The paper is organized as follow. The next section presents a complex robot behavior to model and to adapt. We then present how we can learn such a structured stochastic model. The following section presents how we can use the learned model, followed by a section on results obtained on a real robot (i.e. the learning phase as well as controlling the robot using the learned model). We then conclude the paper with a discussion on the current work as well as some perspectives we are currently pursuing.

2 Modeling and Adapting Robot Behavior

To conduct our experiments, we modeled a robotic navigation task, based on ND² reactive obstacle avoidance [17]. A schematic view of this navigation modality can be seen on figure 1(a). The laser range finder gives a set of points representing obstacles in front of the robot. These points are used to build a local map around the robot by the *aspect* module. ND uses the current position of the robot to compute the local goal in the robot’s coordinate system. Then with this local goal and the map around the robot, it computes a speed reference that tends to move the robot towards the goal position while avoiding obstacles. This modality is used on our B21R robot (figure 1(b)).

² Note that despite the fact that ND was partially developed in our lab (over a number of years), it is so intrinsically complex, and so “difficult” to tune, that no model exists which could help us supervise, control and plan its execution.

¹ LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Cedex 4, Toulouse, France, email: {surname.name}@laas.fr

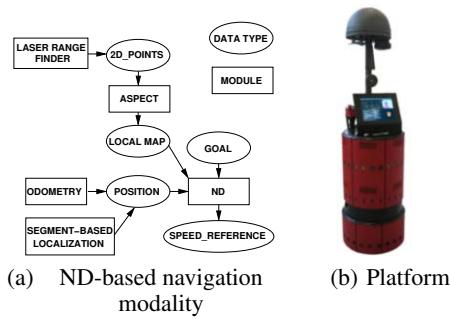


Figure 1. Rackham: An RWI B21R modified to be a museum guide robot [7]

2.1 Structure of a Navigation Task

A navigation controller can be seen as a “black box” taking a relative goal position and some data about the environment as inputs and giving as output a speed reference for the robot to execute. We can model this by having an internal hidden state of the robot, with hidden variables. The control parameters of the navigation are observable and have an influence on the internal state. It changes the environment, because the environment is measured through the sensors of the robot: if the robot goes into a dead-end, the environment will seem very cluttered. The general structure can be seen on figure 2.

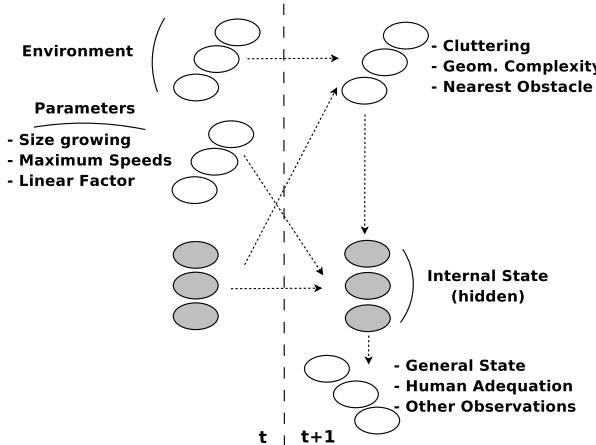


Figure 2. Abstract DBN structure for a navigation task

Choosing the variables is highly dependent on the process to model. The control parameters are given by the navigation controller, but we must choose wisely environment variables that represent important parameters for decisions made by the control process. In our reactive navigation task, the cluttering of the environment is to be taken into account, so is the closest obstacle, which has much influence on the future sight of the robot. On the other hand, we avoid including the position of the robot, so that the model can be used in places different from the one where the learning was performed.

The effects of the navigation task we need to recognize are the *fail* and *success* states, but we also aim at controlling more precisely the behavior in a qualitative way: some behaviors are successful, but not optimal. We also include as many variables as we can that could give us a hint on the internal state, for the learning process to be more

effective. Finally, we could also model resource usage.

2.2 Instantiation

For our navigation task, the control parameters are: the two **size growing** parameters of the robot: one surface represents the robot (where obstacles are totally forbidden); and a larger security area where there should be as few obstacles as possible; the **maximum linear and angular speeds** allowed; a **linearity factor** between the two speeds given by the motion generator controller.³

The environment variables chosen are: the **cluttering** of the environment, defined as a weighted sum of distances to nearest obstacles around the robot; the **angle of the nearest obstacle**; the **distance to the nearest obstacle**; the global **number of segments** in the scene and the number of possible ways to go (**valleys**) built by ND.

The output observable variables are: the current **linear and angular speeds** of the robot; its current **linear and angular accelerations**; the **variation of the distance to the goal** (estimated as an euclidean distance); the **achieved ratio** of the mission; the **current strategy** chosen by ND; the **general state** in *begin, end, fail, normal*; the **human adequacy** of the behavior⁴.

All the variables are discretized (into clusters from 3 to 6 values). The number of different potentially possible observations is more than 15×10^9 . This indicates that even with a very large number of runs, all possible observations will never be seen, leading to many non-informative transition probabilities into the DBN. We will show later how to deal with such a sparse DBN for decision making.

3 DBN Learning

To learn a DBN, we use an adaptation of the classical Expectation-Maximization (EM) algorithm defined for Hidden Markov Models [20]. The adaptations to make this algorithm tractable for DBN are not straightforward: we need to maintain an approximated “belief state”, i.e. a probabilistic hypothesis over the possible assignment of the values of the variables because they are strongly correlated. We choose to approximate this belief state as a particle filter, with a variable number of particles.

3.1 Parameters Learning

A DBN λ is defined by a set of variables (hidden or observable), a set of possible values for each variable, and a set of probabilistic causal links between the variables [8]. An evidence O is as a sequence of observations (i.e. instantiations of all observable variables of λ).

A good model λ with respect to O gives a high $P(O|\lambda)$ (likelihood of the evidence). In the DBN framework, λ may be adapted to the evidence either by modifying its structure or the probabilities attached to the causal links in a given structure. The EM algorithm can deal with hidden variables, but not with structure of the model.

In the HMM case, EM first compute probabilities of being in every hidden state, then every transition probability and every probability of seeing each observation from each hidden state, this for every time step. This is often referred as the Expectation step. For a DBN, the hidden state becomes an instantiation of the corresponding variables, and so does an observation. The state has to be approximated

³ The linear and angular speeds are mutually constrained by a polynomial function that influences the type of trajectory produced.

⁴ Some behaviors leads to success, but may be considered as too aggressive for humans around the robot, or on the contrary too shy, and thus to slow. Its value has to be given by the operator during the learning phase.

as a *belief state*, i.e. a set of weighted hypotheses over all possible instances. Furthermore, the observable variables are not decoupled from the hidden ones, so we compute an observation probability not only from hidden variables, but from the whole belief state. Then we can update the transition probabilities to maximize $P(O|\lambda)$. Then we go again into Expectation and Maximization steps. This algorithm is proved to reach a local maximum of $P(O|\lambda)$.

Choosing a good representation for the belief state is critical for the algorithm to behave correctly. A flat representation over the variables would lose all the correlations among them, being a very bad approximation. On the other side, maintaining an exact belief state implies keeping all correlations over time, and thus is exponentially complex over the length of the evidence [4]. Approximating the belief state with a particle filter seems to be a good trade-off. But while the size of the hypothesis state is very large, we choose to have a variable number of particles, because the probability of seeing the observations knowing the belief state might become very low. We maintain this probability by changing the number of particles in order to have always the same probability of seeing the observation [15].

3.2 Structure Learning

The EM algorithm updates only the transitions probabilities, without changing the structure of the DBN. An algorithm has been proposed [11] to add inside the loop some structural changes. The main issue is to evaluate the new model efficiently knowing the current one. Furthermore, this algorithm acts only as a local search into the structure space, so the optimal structure may not be reached. The GES algorithm [6] has been proposed to find an optimal structure of a Bayesian network. However, this technique relies upon sufficient statistical knowledge of the process, which is not our case due to the very large size of the observation space. In our case, the global structure is known. In order to decrease the size of the DBN and speed up the learning process, we plan to use these techniques only on the hidden variables of the process (and on the links toward the observable variables). For now, we use the structure described in section 5.1.

3.3 Incremental Learning

An open issue for learning stochastic models is the possibility to add knowledge into a previously learned model. This is difficult because the learning phase is a local search, and the algorithm must take into account all observations with the same weight.

We propose the following algorithm: we first learn λ_1 upon a training sequence $(O_1..O_i)$. Then we learn λ_2 over $(O_j..O_n)$, taking λ_1 as a start point for the local search of λ_2 ⁵. When its done, we simply sum λ_1 and λ_2 . For this merging to work, it must be applied after each iteration of EM, i.e. the merged DBN must be used for Maximization and not only for Expectation. We think this is because the role of the hidden variables may be changed otherwise, loosing the “meaning” of them. If the merging is done only between the different phases, the newly learned DBN optimizes itself only in the space of the current trunk of observations and goes somewhat “too far” from the previous one. In this case, the sum of the two DBNs does not reflect the reality of the space of the sum of the observations. This will be investigated in future work.

⁵ Note that λ_1 was learned from a random starting point. Thus the next phases of the learning should be better than the first, because the start point is not random, but somewhat represents a part of the dynamic of the system.

4 Adaptation of the Behavior

One possible use of the model presented in section 2.1 is to find optimal values for the ND motion generator controller, depending on the environment. ND is usually used with a fixed set of parameters, that are not optimal in every situation. We aim at having a fine-grained adaption of the parameters of the controller, modeled as a DBN, to optimize the navigation itself.

We need to introduce utilities into some of our variables with respect to an optimality criterion. We need to give high rewards to variable values that will lead to a desired behavior and penalties to values that will lead to a bad one. Typically, we need to avoid the *fail* value of the **general state** variable, while we want to reach the *end* value.

We also introduce a secondary criterion on the **human adequacy** of the navigation behavior. Between two successful behaviors, we prefer a behavior where the *normal* value of the **human adequacy** appears often, and try to avoid the *aggressive* and *shy* values of this variable. All these utilities will be given as scalar values, so we need to give greater values to the **general state** variable, and smaller ones to the secondary criterion, to avoid the case where the robot could collect reward by having a proper behavior with respect to humans, yet fail, thinking it is better to be gentle than to achieve its task.

4.1 Decision

In this application, the temporal aspect is primordial. The command process works at a frequency of 2.5 Hertz. The behavior adapter has a frequency of 1.66 Hz to collect the observations, but we do not need it to change the parameters at such a high frequency. Typically the parameters changes should occur at most at 0.5 Hertz, or less. Otherwise, the system could demonstrate a very unstable behavior.

So the problem we face is quite different from a classical Dynamic Decision Network [23] resolution, where a decision is taken after each observation, and the result is a decision tree which associates to each observation sequence an optimal decision sequence. Furthermore, the branching factor for a decision tree would be too high. We need a radically different strategy: we consider that when we take a decision, it will remain the same for a given amount of time. So we independently evaluate every decision towards a given horizon, and choose the best one.

The DBN we build includes a model of the evolution of the environment, thus we can accurately predict what will happen in the future. Starting from a belief state for the current time step (i.e. a probabilistic hypothesis on the current state of the system, including hidden variables), we can infer from the DBN the future belief states for each set of parameters and deduce the corresponding utilities.

The belief state is represented as a particle filter, each particle represents a weighted hypothesis on the values of the variables. When inferring the particles over time, we can compute the expected utility by simply summing utilities encountered by particles. This can be done for a few steps forward (which leads to imprecise expectations), or until all particles reach a final state (defined by the *end* and *fail* values of **general state**). The size of the estimation of the belief state (i.e. the number of particles used) is therefore a critical bottleneck for real-time decision making; to solve this issue, we sample the belief state to focus only on the few most probable hypothesis.

4.2 Meta Decision

We introduce a confidence factor in every transition probability. This confidence factor is the number of updates of a transition probability

during the learning phase. This helps us to differentiate an equiprobability due to a never seen transition into the data set from an informative one. When inferring particles to predict expected utilities of the decisions, we also collect the confidence factors of the transitions the particles used. If a particle comes with a high confidence factor, this means that this particle re-played in a learned set, whereas if the confidence factor is low, the particle went through never learned transitions. Thus the decision has to be taken in a two dimensional space: *utility* \times *confidence*. If we prefer to take a decision with high confidence, this will lead to re-play a previously learned observation, and the robot will preferably exploit its model. On the contrary, if we choose a decision with a low confidence factor, this will lead go through never learned transitions of our model, making the robot more “exploratory” of its capabilities, with a higher risk of failure. We encounter here the classic learning agent “exploration versus exploitation” trade-off, in an original fashion.

5 Results

5.1 Navigation Behavior Learning

The model management is implemented into a real-world robotic architecture [1], on a B21R robot. The monitoring of raw data is done at a frequency of 2.5 Hertz. The scalar raw data are clustered using k-means clustering [13] in order to obtain a few clusters.

The a priori adjacency structure of the graph is the following:

- in the same time step, every environment variable is linked to every hidden variable, every hidden variable is connected to every other hidden one, and finally every hidden variable is connect to every post-control observable variable;
- from step t to step $t + 1$, every observable variable is linked to itself, every control variable is linked to every hidden variable, and every hidden variable is connected to itself.

We did about 120 navigations of the robot into different environments, choosing randomly the control parameters at random frequencies; for a total of a few hundred meters of motion into cluttered environments, with people around the robot. The operator was asked to give the value of the **human adequacy** (into *good*, *aggressive*, and *shy*). He also has a *fail* button to stop navigations before dangerous failure like collisions. A failure case was automatically detected when the robot did not move during a given amount of time. The number of observations collected is about 7,000. Yet, it does not cover the total observation space at all. The learning of the DBN takes several minutes per iteration of Expectation-Maximization. The learned DBN stabilizes after less than 10 iterations. As EM is a local search, we use a “conservative” DBN as a starting point, meaning that with this DBN the probability that the variables change their value is low (about 10%).

To evaluate the quality of the learned DBN, we learn the DBN on all observations but one, and try to predict this observation. This is done by inferring a particle filter on the DBN, choosing the most probable observation in this particle filter, and comparing this predicted observation to the actual one. The global observation is seldom exactly the same, but is generally very close, that means that in most cases the particle filter predicts well most of the observable variables. An overview of recognition results is shown on table 3. We can notice that the recognition results are significantly better than a random prediction. Furthermore, if some aspects of the internal mechanisms of ND have not been well modeled (as the strategy choice, or number of valleys build) because of the lack of the environment variables that uses ND “for real”, the general behavior and the human adequacy are

well predicted. And these are the variables we use in particular for control.

variable	random guess	dbn
cluttering	33.3	87.3
angle of obstacle	25.0	86.4
dist. to obstacle	25.0	88.5
# segments	33.3	81.6
# valleys	33.3	79.9
v	33.3	90.8
w	25.0	79.0
dv	33.3	91.9
dw	33.3	92.2
Δ dist. to goal	33.3	77.8
% mission	33.3	97.1
strategy	20.0	84.8
general	25.0	96.4
adequacy	33.3	92.5
average	29.0	87.6

Figure 3. Table of well-guessed one-step predictions (%)

Using more hidden variables gives better results, meaning that 2 hidden variables with an arity of 2 is not enough. But using more hidden variables makes the learning process much longer because of the “complete” structure we use. Here we consider 49 causal links, while with 3 variables it comes up to 73. We intend to decrease this amount of links using structure learning in future work.

5.2 Control

The decision level of the behavior adapter was implemented using OpenPRS [12]. The current belief state is maintained along the navigation, and when a choice is to be made, it is sampled in order to limit the size of the particle filter for prediction.

In our control experiments, the choices were made at a fixed period of 2 seconds, changing reactive obstacle avoidance parameters dynamically during navigation, without stopping the robot motion. Therefore, the navigation behaves better than with default static hand-tuned parameters usually used on this robot, and much better than with random parameters. Typically, during the learning phase with random parameters, 1 run out of 3 ends on a failure state due to collisions or to blocked situations, while with the behavior adapter, the failures happen only once out of 10 experiments. This happens while using a parameter set never tried during the learning phase⁶. Also, in the learning phase, we label as *good* the parameters producing very high accelerations when the robots enters an open area. We give rewards to this value for decision making, and this behavior is clearly noticeable in these experiments, while it is not with the default hand-tuned parameters. This gives us quicker navigation for open areas than with default parameters. The average speedup of the navigation will be quantified in a future work.

Collecting observations can be made in the same time than active behavior adapting; the next step is to decide when the robot should use computed behavior or behaviors that extend the variety of the set of collected observations.

⁶ This is due to a confidence factor allowing learning more than using the model.

5.3 Incremental Learning

To evaluate the incremental learning algorithm, we learned a DBN in five phases as explained in section 3.3. Each phase uses 20% of the total observation. On figure 4, the phases are numbered on the X-axis. The Y-axis represents the log-likelihood obtained. The upper line shows the log-likelihood obtained with a single phase of learning (upon all observations). While the summed DBN does not behave as well as the one learned in a single phase, the difference is very small.

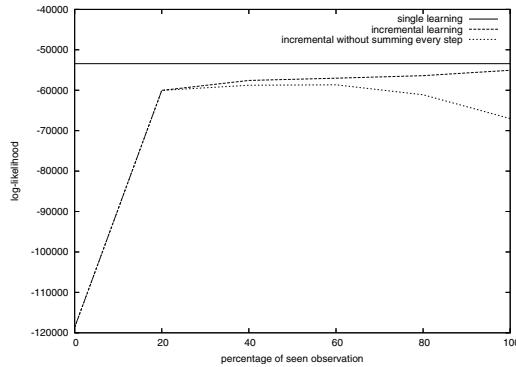


Figure 4. Incremental learning likelihoods

6 Conclusion and Discussion

We have presented an innovative approach to learn DBN models of complex robot behaviors. In our context, where we want to learn a model where controllable variables remain observable, DBN are preferable to HMM. We developed a modified version of EM for DBN. We presented our behavior controller, and how it can be used to learn the model and then use it online. The approach has been implemented and successfully tested on a robotic platform. We showed that, for a fairly complex navigation task, we were able to learn a model which, when used to dynamically set the controllable parameters (according to given objectives), performs much better than the default set, or a random set.

This work is closely related to [16], where the authors tries to model human activities with hierarchical DBN. Our approach tries to deal with more variables, and with sparse data sets. Furthermore, we propose a direct way to use our model not only for diagnosis, but to control the process itself. In [18], the authors learn to recognize robotic behaviors that fits with human judgment, but the emphasis is more on “how to built variables that make a intuitive sense” than on use of this kind of model. The proposed approache is a more classical planning one [21]. We propose an efficient way for mixing learning and decisions making. Our approach is in the same spirit than [3], where the authors model the system as neural nets or with tree induction, then take decision based on a MDP, for robotic autonomous navigation. We propose a more integrated approach, where the decision-taking mechanism is much closer from the model itself. Furthermore, it allows taking into account hidden variables, i.e. a more complex structure of the model itself. We propose a way to build a model using more features than the one proposed. In [22] the authors present an original way of optimizing tasks execution. High-level tasks are built with a number of parameters that must be optimized for smooth execution. Performance models are learned for the parameter sets as tree rules, but much of the work is hand-made. Our approach is automated, and the models we build are more expressive (while harshly

human-readable), due to the presence of hidden variables. Furthermore, the models we built are fully probabilistic, while this is not the case in this work. Both approaches show very high computational cost when different complex tasks have to be mixed.

In the long run, we keep in mind that these learned behaviors should not only be used to control the robot execution (to avoid failure and to optimize the execution), but can also be taken into account by a high level planner able to choose among a set of such behaviors.

Despite our application to a navigation task of a museum guide robot, we believe that the approach remains applicable to other behaviors and could be used for other autonomous systems.

REFERENCES

- [1] R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morisset, P. Moutarlier, and T. Simon, ‘Around the lab in 40 days...’, in *Proceedings of ICRA*, (2000).
- [2] E. Amir, ‘Learning partially observable deterministic action models’, in *Proceedings of IJCAI*, (2005).
- [3] T. Belker, M. Beetz, and A. Cremers, ‘Learning action models for the improved execution of navigation plans’, *Robotics and Autonomous Systems*, **38**(3-4), 137–148, (2002).
- [4] X. Boyen and D. Koller, ‘Tractable inference for complex stochastic processes’, in *Proceedings of UAI*, (1998).
- [5] H. Bui, ‘A General Model for Online Probabilistic Plan Recognition’, in *Proceedings of IJCAI*, (2003).
- [6] D. Chickering, ‘Optimal structure identification with greedy search’, *Journal of Machine Learning Research*, **3**, 507–554, (2002).
- [7] A. Clodic, S. Fleury, R. Alami, M. Herrb, and R. Chatila, ‘Supervision and interaction’, in *Proceedings ICAR*, pp. 725–732, (2005).
- [8] T. Dean and K. Kanazawa, ‘A model for reasoning about persistence and causation’, *Computational Intelligence*, **5**(3), 142–150, (1990).
- [9] A. Dearden and Y. Demiris, ‘Learning forward models for robots’, in *Proceedings of IJCAI*, (2005).
- [10] M. Fox, M. Ghallab, G. Infantes, and D. Long, ‘Robot introspection through learned hidden markov models’, *Artificial Intelligence*, **170**(2), 59–113, (february 2006).
- [11] N. Friedman, ‘The bayesian structural EM algorithm’, in *Proceedings of UAI*, (1998).
- [12] F. Ingrand, R. Chatila, R. Alami, and F. Robert, ‘PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots’, in *Proceedings of ICRA*, (1996).
- [13] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu, ‘An efficient k-means clustering algorithm: analysis and implementation’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(7), 881–892, (July 2002).
- [14] S. Koenig and R. G. Simmons, ‘Unsupervised Learning of Probabilistic Models for Robot Navigation’, in *Proceedings of ICRA*, (1996).
- [15] D. Koller and R. Fratkina, ‘Using learning for approximation in stochastic processes’, in *Proceedings of ICML*, (1998).
- [16] L. Liao, D. Fox, and H. Kautz, ‘Learning and Inferring Transportation Routines’, in *Proceedings of AAAI*, (2004).
- [17] J. Minguez, J. Osuna, and L. Montano, ‘A “Divide and Conquer” Strategy based on Situations to achieve Reactive Collision Avoidance in Troublesome Scenarios’, in *Proceedings of ICRA*, (2004).
- [18] T. Oates, M. D. Schmill, and P. R. Cohen, ‘A method for clustering the experiences of a mobile robot that accords with human judgements’, in *Proceedings of IJCAI*, (2000).
- [19] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, Morgan Kaufmann, 1988.
- [20] L. R. Rabiner, ‘A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition’, *Proceedings of the IEEE*, **77**(2), 257–286, (February 1989).
- [21] Matthew D. Schmill, Tim Oates, and Paul R. Cohen, ‘Learning planning operators in real-world, partially observable environments’, in *Proceedings of ICAPS*, (2000).
- [22] F. Stulp and Michael Beetz, ‘Optimized execution of action chains using learned performance models of abstract actions’, in *Proceedings of IJCAI*, (2005).
- [23] N.L. Zhang, R. Qi, and D. Poole, ‘A computational theory of decision networks’, *International Journal of Approximate Reasoning*, **11**(2), 83–158, (1994).

Plan-Based Configuration of a Group of Robots

Robert Lundh and **Lars Karlsson** and **Alessandro Saffiotti**¹

Abstract. We consider groups of autonomous robots in which robots can help each other by offering information-producing functionalities. A functional *configuration* is a way to allocate and connect functionalities among robots. In general, different configurations can be used to solve the same task, depending on the current situation. In this paper, we define the idea of functional configuration, and we propose a plan-based approach to automatically generate a preferred configuration for a given task, environment, and set of resources. To illustrate these ideas, we show a simple experiment in which two robots mutually help each-other to cross a door.

1 Introduction

Consider the situation shown in Figure 1, in which a mobile robot, named Pippi, has the task to push a box across a door. In order to perform this task, Pippi needs to know the position and orientation of the door relative to itself at every time during execution. It can do so by using its sensors, e.g., a camera, to detect the edges of the door and measure their distance and bearing. While pushing the box, however, the camera may be covered by the box. Pippi can still rely on the previously observed position, and update this position using odometry while it moves. Unfortunately, odometry will be especially unreliable during the push operation due to slippage of the wheels. There is, however, another solution: a second robot, called Emil, could observe the scene from an external point of view in order to compute the position of the door relative to Pippi, and communicate this information to Pippi.

The above scenario illustrates a simple instance of the general approach that we suggest in this paper: to let robots help each other by borrowing functionalities from one another. In the above example, Pippi needs a functionality to measure the relative position and orientation of the door in order to perform its task: it has the options to either compute this information using its own sensors, or to borrow this functionality from Emil.

More generally, we consider a society of autonomous robotic systems embedded in a common environment [5]. Each robot in the society includes a number of *functionalities*: for instance, functionalities for image understanding, localization, planning, and so on. We do not assume that the robots are homogeneous: they may have different sensing, acting, and reasoning capacities, and some of them may be as simple as a fixed camera monitoring the environment. The key point here is that each robot may also use functionalities from other robots in order to compensate for the ones that it is lacking, or to improve its own. In the situation shown in Figure 1, Pippi borrows from Emil a perceptual functionality for measuring the relative position between the door and itself.

¹ Center for Applied Autonomous Sensor Systems, Örebro University, Sweden. email: {robert.lundh, lars.karlsson, alessandro.saffiotti}@aass.oru.se

We informally call *configuration* any way to allocate and connect the functionalities of a distributed multi-robot system. Often, the same task can be performed by using different configurations. For example, in our scenario, Pippi can perform its door-crossing task by connecting its own door-crossing functionality to either (1) its own perception functionality, (2) a perception functionality borrowed from Emil, or (3) a perception functionality borrowed from a camera placed over the door. Having the possibility to use different configurations to perform the same task opens the way to improve the flexibility, reliability, and adaptivity of a society of robotic agents. Ideally, we would like to automatically select, at any given moment, the best available configuration, and to change it when the situation has changed.

In this context, our overall research objective is threefold:

1. To formally define the concept of functional *configuration* of a robot society.
2. To study how to *automatically generate* a configuration of a robot society for a given task, environment, and set of resources.
3. To study when and how to *change* this configuration in response to changes in the environment, task, or resources.

In this paper, we focus on the first two objectives above: the third objective will be the subject of future work. More specifically, we define a concept of configuration which is adequate from the purpose of automatically reasoning about configurations, and show how to use AI planning techniques to generate a configuration that solves a given task. We also describe an experimental system where these ideas are

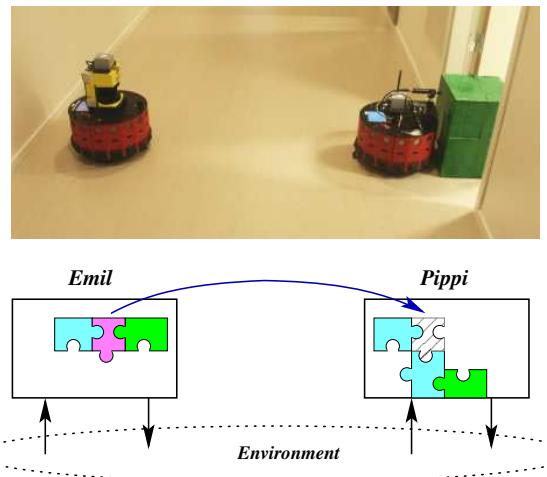


Figure 1. Top: Emil helps Pippi to push the box through the door by measuring their relative position. Bottom: the corresponding configuration in which Emil is providing a missing perceptual functionality to Pippi.

implemented, and show an example of it in which two iRobot Magellan Pro robots mutually help each other to cross a door.

2 Functional configurations

The first goal in our research program is to develop a definition of configuration that is adequate for the three objectives presented in the introduction. In general, a configuration of a team of robots may include interconnected functionalities of two types: functionalities that change the internal state by providing or processing information, and functionalities that change the state of the environment. (Some functionalities can have both aspects.)

To define our notion of configurations, a clarification of the three concepts of functionality, resource and channel is in order.

2.1 Preliminaries

We assume that the world can be in a number of different states. The set of all potential world states is denoted S . There is a number of robots r_1, \dots, r_n . The properties of the robots, such as what sensors they are equipped with and their current positions, are considered to be part of the current world state s_0 . There are also a number of communication media CM , such as radio, network, internal message queues, which can be used to transfer information between and within robots. A medium may have restrictions on band width.

2.2 Functionality

A *functionality* is an operator that uses information (provided by other functionalities) to produce additional information. A functionality is denoted by

$$f = \langle r, Id, I, O, \Phi, Pr, Po, Freq, Cost \rangle$$

Each instance of a functionality is located at a specific robot or other type of agent r and has a specific identifier Id .² The remaining fields of the functionality tuple represents:

- $I = \{i_1, i_2, \dots, i_n\}$ is a specification of inputs, where $i_k = \langle descr, dom \rangle$. The descriptor (*descr*) gives the state variable of the input data, and the domain (*dom*) specifies to which set the data must belongs.
- $O = \{o_1, o_2, \dots, o_m\}$ is a specification of outputs, where $o_k = \langle descr, dom \rangle$. The descriptor (*descr*) gives the state variable of the output data, and the domain (*dom*) specifies to which set the data must belongs.
- $\Phi : dom(I) \rightarrow dom(O)$ specifies the relations between inputs and outputs.
- $Pr : S \rightarrow \{T, F\}$ specifies the causal preconditions of the functionality. Pr specifies in what states $s \in S$ the functionality can be used.
- $Po : S \times dom(I) \rightarrow S$ specifies the causal postconditions. It is a function that, given the input to the functionality, transforms the world state s before the functionality was executed into the world state s' after the functionality has been executed.
- $Freq$ specifies how often the functionality is to be executed.
- $Cost$ specifies how expensive the functionality is, e.g. in terms of time, processor utilization, energy, etc.

² When referring to a specific element in a functionality or other entity represented as a tuple, we will be using a functional notation, e.g. $r(f)$ is the r field in the tuple of f .

A typical functionality could be the measure door operation mentioned in the introductory example. This functionality takes an image from a camera as an input and measures the position and orientation of a door in the image. To produce the output, the position and the orientation of the door, this functionality has a precondition that needs to be satisfied. The precondition is that the door must be visible in the (input) image.

2.3 Resource

A *resource* is a special case of a functionality. There are two different types of resources: sensing resources and action resources. A *sensing resource* has $I = \emptyset$, i.e., no input from other functionalities, and is typically a sensor that gives information about the current state of the surrounding environment or the physical state of the robot. An example of a sensing resource is a camera, which produces images as output as long as the preconditions (e.g. camera is on) are fulfilled.

An *action resource* has $O = \emptyset$ (i.e., gives no output to other functionalities) and Po is not the identity function. An action resource is typically some actuator (e.g., a manipulator).

2.4 Channel

A *channel* $Ch = \langle f_{send}, o, f_{rec}, i, medium \rangle$ transfers data from an output o of a functionality f_{send} to an input i of another functionality f_{rec} . It can be on different media.

2.5 Configuration

A *configuration* C is tuple $\langle F, Ch \rangle$, where F is a set of functionalities and Ch is a set of channels that connect functionalities to each other. Each channel connects the output of one functionality to the input of another functionality.

In the context of a specific world state s , a configuration is *admissible* if the following conditions are satisfied:

$$\forall f \in F \forall i \in I_f \exists ch \in Ch = \langle f_{send}, o, f_{rec}, i, m \rangle \text{ such that } descr(o) = descr(i), dom(o) = dom(i), \text{ and } Freq(f_{send}(ch)) \geq Freq(f_{rec}(ch))$$

That is, each input of each functionality is connected via an adequate channel to an output of another functionality with a compatible specification (information admissibility).

$$\forall f \in F : Pr_f(s) = T$$

That is, all preconditions of all functionalities hold in the current world state (causal admissibility).

$$\forall m \in CM : bandwidth(m) \geq \sum_{\{ch | medium(ch)=m\}} size(domain(i(ch))) \cdot Freq(f_{rec}(ch))$$

That is the combined requirements of the channels can be satisfied (communication admissibility).

Another issue is schedulability: whether the different functionalities can produce and communicate their outputs in a timely manner. However, that is a complex issue that cannot be detailed in the scope of this paper.

A configuration also has a cost. This can be based on functionality costs, communication cost, etc, but also on performance accuracy and reliability of the configuration. Currently, we compute the cost as a weighted sum of the number of components used (robots involved, functionalities, global and local channels).

2.6 Examples

To illustrate the above concepts, we consider a concrete example inspired by the scenario in the introduction. A robot is assigned the task of pushing a box from one room to another one by crossing a door between the two rooms. The “cross-door” action requires information about position and orientation of the door with respect to the robot performing the action. There are two indoor robots (including the one crossing the door) each one equipped with a camera and a compass. The door to cross is equipped with a wide-angle camera.

Fig. 2 illustrates four different (admissible) configurations that provide the information required by the action “cross-door”, which include the functionalities above.

The first configuration involves only the robot performing the action. The robot is equipped with an elevated panoramic camera that makes it possible to view the door even when pushing the box. The camera produces information to a functionality that measures the position and orientation of the door relative to the robot.

The second configuration in Fig. 2 shows the other extreme, when all information is provided by the door that the robot is crossing and the robot is not contributing with any information. The door is equipped with a camera and functionalities that can measure the position and orientation of the robot relative to the door. This information is transformed into position and orientation of door with respect to the robot before it is delivered to robot A.

In the third configuration in Fig. 2, robot A (the robot performing the “cross-door” action) contributes with a compass, and the second robot (*B*) contributes with a compass and a camera. The camera provides information to two functionalities: one that measures the position and orientation of the door, and another one that measures the position of robot A. All these measurements are computed relative to robot B. In order to compute the position and orientation of the door relative to robot A, we need to use a coordinate transformation. This in turn requires that we know, not only the relative position, but also the orientation of robot A relative to B. The later can be obtained by comparing the absolute orientations of the two robots, measured by their two on-board compasses.

The fourth configuration in Figure 2 is similar to the third one, except that the orientation of robot A relative to B is obtained in another way, i.e., no compasses are used. Both robots are equipped with cameras and have a functionality that can measure the bearing to an object. When the robots look at each other, each robot can measure the bearing to the other one. By comparing these two measurements, we obtain the orientation of robot A relative to robot B.

3 Configuration generation

Our second objective is to generate configurations automatically. This process requires a declarative description of:

- the functionalities and methods for connecting them,
- the current world state,
- the goal for what the configuration should produce.

The world state, as already mentioned, declares the available robots and their physical capabilities, as well as the state of the surrounding environment. It is encoded as a set of clauses, such as `robot(r1)`, `door(d1)`, `visible(r1, door1)`. The goal for the configuration generation process is to produce a desired information output. For example, in the cross-door example, the information goal is the position and orientation of the door that is required as input by the cross-door behavior.

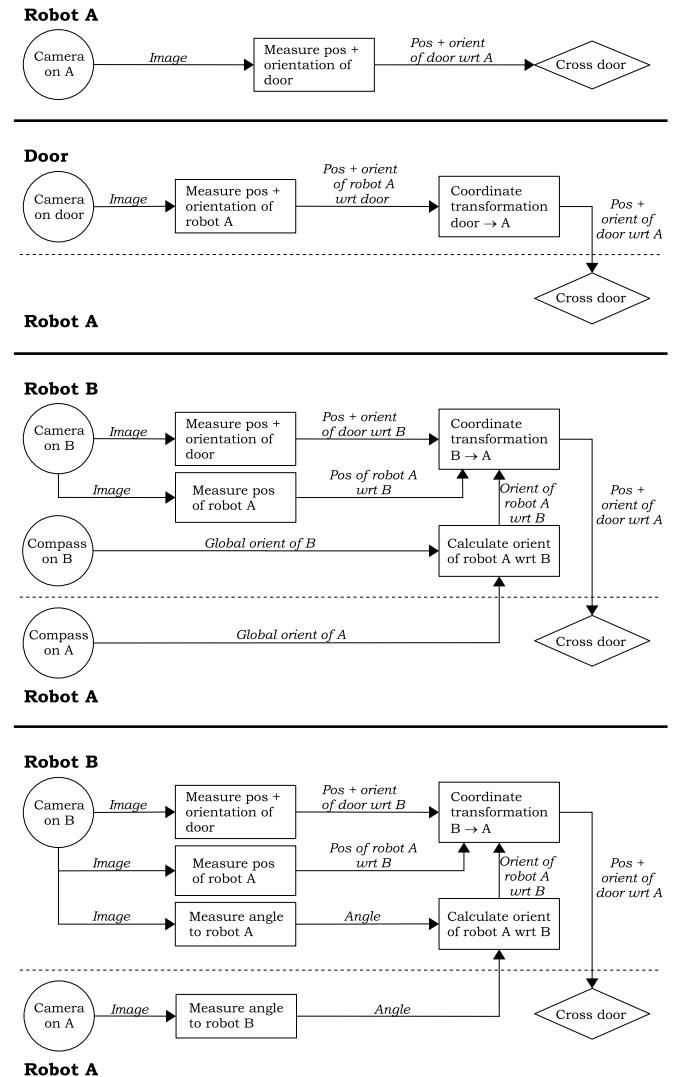


Figure 2. Four different configurations that provide the position and orientation of a given door with respect to robot A. (See text.)

The description of available functionalities is realized using operator schemas similar to those of AI action planners. One functionality operator schema from the scenario in the previous section, `measure-door`, is shown in the upper half of Figure 3.

The name field correspond to the *Id* component of a functionality, and the *r* parameter in the name is *r* in the functionality. The fields *input* and *output* correspond to *I* and *O*, and *precond* and *postcond* encode the *Pr* and *Po* components. In the `measure-door` example we have an image taken by the camera of *r* as input and from that we are able to compute the position and orientation of the door *d* relative to *r* as output. The precondition for `measure-door` is that the door *d* is fully visible in the input image. Note that the Φ function is not part of the operator schema; it corresponds to the actual code for executing the functionality.

In order to combine functionalities to form admissible configurations that solve specific tasks, we have chosen to use techniques inspired by hierarchical planning, in particular the SHOP planner [3]. Configuration planning differs from action planning in that the

```

(functionality
  name: measure-door(r, d)
  input: (image(r, d), image)
  output: (pos(r, d), real-coordinates)
           (orient(r, d), radians)
  precond: visible(r, d)
  postcond:
)

(config-method
  name: get-door-info(r, d)
  precond: (camera(r), in(r, room),
            in(d, room), robot(r), door(d))
  in: -
  out: f2: pos(r, d)
       f2: orient(r, d)
  channels: (f1, f2, image(r, d))
  body:
    f1: camera(r, d)
    f2: measure-door(r, d)
)

```

Figure 3. A functionality operator schema, and a method schema for combining functionalities.

functionalities, unlike the actions in a plan, are not temporally and causally related to each other, but related by the information flow as defined by the channels. Functionalities are executed in parallel, and a functionality becomes active when the input data is present. In order to achieve the data flow between functionalities, a mechanism that creates channels between functionalities is required. Such connections are not required for action planning, and obviously, no existing planning technique facilitate such a mechanism. Therefore we need to outline how the planner works.

Our configuration planner allows us to define methods that describe alternative ways to combine functionalities (or other methods) for specific purposes, e.g. combining the camera functionality and the measure-door functionality above with a channel on the same robot in order to obtain door measurements of a specific door.

The lower half of Figure 3 shows an example of a method schema that does exactly that. There is a channel inside the method connecting two functionalities (labeled f_1 and f_2). The descriptor of the channel ($image(r, d)$) tells which specific input and output of the functionalities should be connected. In addition, the outputs ($pos(r, d)$, $orient(r, d)$) of f_2 is declared in the out field to be the output of the entire method. Thereby, any channel that in a method higher up in the hierarchy is connected to the output of $get-door-info$ will be connected to the output of $measure-door$.

The configuration planner takes as input a current causal state s , a method "stack" with initially one unexpanded method instance $l : m(c_1, c_2, \dots)$ representing the goal of the robot (l is a label), and a set of methods M and a set of functionality operators O . It also maintains an initially empty configuration description C and an initially empty sets of causal postconditions P . It works as follows:

1. Take the unexpanded method instance $l : m(c_1, c_2, \dots)$ at the top of the stack.
2. If $l : m(c_1, c_2, \dots)$ matches the name of a functionality operator O , check if that functionality already exists in the current configuration description C . If it does not, add it to C , and add the clauses in $postcond$ to P . If there is an $l' : m(c_1, c_2, \dots)$ reuse it and reconnect any channel from/to l from/to l' instead. Go back to 1.
3. If $l : m(c_1, c_2, \dots)$ matches a method schema in M , select an instantiated version of the method schema from M with preconditions which are applicable in s (this is a backtrack point). If there

is none, report failure and backtrack.

4. Expand the method schema as follows:

- (a) Instantiate the remaining fields of the selected method, and generate new unique labels instead of the general labels f_1, f_2 etc.
- (b) Add the channels of the method body (with new labels) to the current configuration C .
- (c) Add the method instances (with new labels) of the method body to the top of the stack.
- (d) Use the in and out fields of the method to reconnect any channels in C from the method instance being expanded (i.e. with label l) to the new method instances as labeled in the method body.

5. If the stack is empty, return C and P . Otherwise go back to 1.

Step 4 may need some more explanation. Assume we are expanding 15: $get-door-info(robot1, door4)$, and we have chosen the method in Figure 3. First, we need to replace r with $robot1$ and d with $door4$ everywhere in the schema, and $room$ is bound to the room of $robot1$ (step a). New labels, say 17 and 18, replace f_1 and f_2 (step a). The channel is added to C (step b) and the two functionalities are added to the top of the stack (step c). Finally, according to the out field, any existing channel in C with label 15 (i.e. $get-door-info$) for its out connection is reconnected to 18 (i.e. $measure-door$) (step d).

The first output from the planner is a configuration description C , which essentially consists of a set of functionality names with labels, e.g. 18: $measure-door(robot1, door4)$, and set of channels, e.g. (17, 18, $image(robot1, door4)$, $local(robot1)$).

The second output from the planner is the set of postconditions P , which can be used to update the current state, which then in turn can be used as input for generating the configuration following the current one (if any).

It is possible to accidentally specify methods that can result in configurations with cycles. However, these cycles can easily be detected and the faulty configurations can be excluded.

Generally, there are several configurations that can address a problem, but obviously, only one configuration per problem can be performed at the time. By trying different applicable method versions, guided by the cost of configurations, our planner generates the admissible configuration with the lowest cost first.

The planning takes place on the robot that has the information goal. The selected configuration description is then implemented in two steps. First, the different functionalities and channels are distributed according to their location parameter. Then each robot launches a process for executing its assigned functionalities and sets up its channels.

4 Experiments

We have implemented the approach described above, and we have conducted a series of experiments using real robots equipped with different sensors. We report here a representative experiment based on the third and fourth configurations in Figure 2. The platforms used were two Magellan Pro robots from iRobot, shown in Figure 1. Both robots are equipped with compasses and fixed color cameras. The environment consists of two rooms (R1 and R2) with a door connecting them. The door and the robots have been marked with uniform colors in order to simplify the vision task.

The following scenario describes how the two configurations were used, and demonstrates the importance of being able to reconfigure dynamically. Robot *A* and robot *B* are in room R1. Robot *A* wants to go from room R1 to room R2. Since the camera on *A* is fixed and it has a narrow field of view, the robot cannot see the edges of the door when it is close to it. Therefore, robot *A* is not able to perform the action on its own. Robot *B* is equipped with the same sensors as robot *A*, but since robot *B* is not crossing the door it is able to observe both the door and robot *A* from a distance during the whole procedure. We therefore configure our team according to the third configuration in Figure 2, and execute the task. Robot *A* continuously receives information about the position and orientation during the execution of “cross-door”.

When robot *A* enters room R1 it signals that the task is accomplished. This signal is received by robot *B* and the current configuration is played out. Next, robot *B* is assigned the task of going from room R1 to room R2. The same configuration as before is used to solve this task, but with the roles exchanged — i.e., robot *A* is now guiding robot *B*. This time, however, during the execution of the “cross-door” behavior a compass fails due to a disturbance in the magnetic field. This makes the current configuration not admissible, and a reconfiguration is necessary to proceed. The fourth configuration in Figure 2 is still admissible even with no compass, and we therefore use this one to carry out the remaining part of the task.

5 Discussion

We have presented a general approach to automatically synthesize a team configuration using knowledge-based techniques. Our approach combines resources and functionalities, residing in different robots, into a functional configuration, in which the robots cooperate to generate the information required to perform a certain task.

Even though the example that we use in this paper to describe our approach is simple, our system is able to generate configurations for more complex problems. For instance, we have also used our configuration planner to address two object transportation tasks. The first task is considering two robots carrying a bar. The second tasks involves three robots building a wall with several wall blocks. In this task, two robots are pushing a wall block together and a third robot is guiding them in order to get the block aligned with the rest of the wall. Both tasks require tight coordination between the involved robots.

An interesting point to note is the relation between our functional configurations and task allocation/assignment [1]. Task allocation typically deals with the question: “Who should do which task?”. That is enough for tasks that only require loose coordination. For tasks that require tight cooperation (that is, they cannot be neatly divided among robots or modules), we must address the additional question: “How to execute a task in a cooperative manner?”. Configuration generation answers this question and can be seen as a proceeding step to task allocation, done after a task is assigned to a robot.

A related problem is how to motivate the robots involved in a configuration to commit to it, and to stay committed during the entire execution. For example, in the experiment above, we must guarantee that the observing robot does not run away from the scene. For most tasks, robots are motivated by the individual or the team goals. However, if the task is beneficial only to another individual, as in the case of robots that help each other, it may be harder to motivate a commitment. Task assignment, commitment and functional configurations all address different aspects of the multi-robot cooperation problem. In our work, we concentrate on the last one.

Problems similar to the work on automatic generation of configurations have been studied in several different research areas, e.g. in program supervision [7], automatic web service composition [4], coalition formation [6], and single robot task performance [2]. However, in the field of cooperative robotics, few works address similar problems. Tang and Parker [8] present an approach called ASyMTRe. The principle of ASyMTRe is to connect different schemas (similar to instantiated functionalities) in a way such that a robot team is able to solve tightly-coupled tasks by information sharing. ASyMTRe uses a greedy algorithm that starts with handling the information needs for the less capable robots and continues in order to the most capable robots. For each robot that is handled, the algorithm tries to find the robots with least sensing capabilities and maximal utility to provide information.

In contrast to the ASyMTRe approach, the approach presented in this paper uses a hierarchical planner to automatically generate configurations. We expect that the use of a hierarchical planner will make it easier to deal with the problem of when and how to change (replan) a configuration. This problem is related to the third objective stated in the introduction: how to monitor the performance of a configuration while executing it. We also believe that the use of a hierarchical planner will be beneficial for the next important step, to consider sequences, or plans, of configurations, in order to address more complex tasks. Our current system only considers the generation of configurations for performing one step of a particular task, and cannot deal with situations requiring several steps. In our box pushing example, if a second box is blocking the door, a configuration for removing that box would have to precede the configuration for getting the first box through the door. We are investigating the extension of our planning techniques to generate plans involving sequences of configurations.

Acknowledgments

This work was supported by the Swedish National Graduate School in Computer Science (CUGS), the Swedish Research Council (Vetenskapsrådet), and the Swedish KK Foundation.

REFERENCES

- [1] B. Gerkey and M. Matarić, ‘A formal analysis and taxonomy of task allocation in multi-robot systems’, *International Journal of Robotics Research*, **23**(9), 939–954, (September 2004).
- [2] B. Morisset, G. Infante, M. Ghallab, and F. Ingrand, ‘Robel: Synthesizing and controlling complex robust robot behaviors’, in *Proceedings of the Fourth International Cognitive Robotics Workshop, (CogRob 2004)*, pp. 18–23, (August 2004).
- [3] D. Nau, Y. Cao, A. Lothem, and H. Munoz-Avila, ‘SHOP: simple hierarchical ordered planner’, in *IJCAI*, (1999).
- [4] J. Rao and X. Su, ‘A survey of automated web service composition methods’, in *In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, San Diego, California, USA, (July 2004).
- [5] A. Saffiotti and M. Broxvall, ‘PEIS Ecologies: Ambient intelligence meets autonomous robotics’, in *Proc of the Int Conf on Smart Objects and Ambient Intelligence (SoC-EUSA)*, pp. 275–280, Grenoble, France, (2005). www.aass.oru.se/~peis/.
- [6] O. Shehory and S. Kraus, ‘Methods for task allocation via agent coalition formation’, *Artificial Intelligence*, **101**, 165–200, (1998).
- [7] C. Shekhar, S. Moisan, R. Vincent, P. Burlina, and R. Chellappa, ‘Knowledge-based control of vision systems’, *Image and Vision Computing*, **17**, 667–683, (1998).
- [8] F. Tang and L. Parker, ‘Coalescing multi-robot teams through ASyMTRe: A formal analysis’, in *Proceedings of IEEE International Conference on Advanced Robotics (ICAR)*, (July 2005).

This page intentionally left blank

III. Posters

This page intentionally left blank

1. Cognitive Modeling

This page intentionally left blank

Agents with Anticipatory Behaviors: To be Cautious in a Risky Environment¹

Cristiano Castelfranchi, Rino Falcone and Michele Piunti²

Abstract. This work presents some anticipatory mechanisms in an agent architecture, modeling affective behaviours as effects of surprise. Through experiment discussion, the advantages of becoming cautious are presented and is shown how this approach increases not only opportunism and reactivity but also anticipatory capabilities for planning and reasoning. Cautious agents outcomes are analyzed in risky surroundings both on the basis of the environment features and of the internal model of caution. Expectation-driven and caution enabled agents are designed with the BDI paradigm.

1 CAUTIOUS AGENTS

This work is based on a general research framework whose objective is to claim how important is for an autonomous, cognitive system to exploit anticipatory behaviors, and in particular anticipatory representations of the world (predictions and expectations). Here we introduce anticipatory mechanisms in a BDI architecture, and model part of the short-term functions of surprise, in particular the advantages (in some kinds of environment) of becoming cautious. To have anticipatory representations, to experience surprise, and to exploit various functions of them will be crucial, not only for curiosity and exploration [5], but also for learning, attention, prudently adjusting behaviour, and for social relations: like trust [2], suspicion; coordination with the others and reliance; cooperation and competition etc. Without examining deep surprise sources and dynamics, we exploit some effects of surprise and their possible outcomes: in risky, unknown, harmful environments agent cautiousness can be a consequence of a kind of surprise that is due to -and a signal of- a mismatch between the agent expectation (internal state) and the actual perceived input (perceived data) [3]. Its internal signal alerts the agent and bring to mental and behavioural changes both with *Long term effects* like becoming more accurate and less self-confident in predictions, memorizing anticipatory signs of danger and different kinds of environment, learning more or less safe plans and purposive actions for a given goal in a certain condition and *Short term effects* like intention reconsideration, redirecting attention, searching for additional information, or becoming prudent. Hence, the anticipatory dimension of caution is strictly related to a form of expectation: like agents use and represent expectations to select best expected action from a set of available ones, so caution guides agent in the selection of less risky action between possible ones. Given this, it is possible to characterize agents with stable 'personalities' (a watchful and a rash

agent, with locked capabilities [7]) and to experiment their performances in more or less dangerous environment. Moreover it is possible to build an adaptive agent able to run-time adjust its "degree" of caution to different environments or situations. This approach enhances anticipatory competences and increases the opportunism and the reactivity of BDI-like [6], planning and deliberative agents [1]. We do assume that the relevance of this problem is not only for theoretical modeling and for playing, but that will soon become important in virtual worlds and in the physical one, both for software cognitive agents and for autonomous robots.

2 SCENARIO DESIGN

A test bed scenario is designed as a simulation where entities and environment are represented as autonomous agents and artefacts. Environment is a 2D land map where sets of walls and gates (that can be open or closed) delimit rooms, corridors and areas where entities are able to move. A set of special locations seat symbolic reference points: these location of interest (LOI) contain noticeable world objects as *Repository*, *House*, *Tree*; two kinds of *Food* objects appear with modifiable frequencies, near to House and Tree: they rise at fixed location with a modifiable "reward" value; Harmful *Frozen puddles*, are located across rooms; *Fires* entities behave according to a two state cycle periodic function: in each period their first shape is a "Smoke" premonitory state, then they become real dangerous "Fire". For each period, fires change their location with discrete movements. Environment holds regularities as delimited risky areas where rise of fires can be related with agent presence.

At an high level of abstraction, we consider agents as mobile entities with sensors, reasoning and effectors components, characterized by the following tuple of bounded dynamic resources:

$$Ag = \langle En, r, Sr, s \rangle \quad (1)$$

En indicating the instant amount of energy, *r* the range of vision where sensors can retrieve data, *Sr* the sensor sample rate, and *s* the agent instant speed. Agents burn energy according to a combination of previous resource allocation (e.g. the more speed and sensor-rate is high, the more agent will spend energy) and they hold structured data sets representing knowledge and internal state.

In a Goal oriented fashion, first order objective for entities is to collect food items in the repository to obtain rewards and recharge energy. The task is composed of a recursive workflow of actions: 1) Look for Food with (supposed) best reward. Because agents know that each class of food rises near a proper LOI, this introduces agent explicit quantitative expectations about the possible food presence (and value). Each *lookForFoodPlan* is associated to a specific LOI and constantly monitored by a *subjective expected utility* function,

¹ This work is supported by the EU project MindRACES: From Reactive to Anticipatory Cognitive Embodied Systems. Contract Number:FP6-511931. See www.mindRaces.org.

² ISTC - Institute of Cognitive Sciences and Technologies - C.n.r. - National Research Council, Italy, email: {c.castelfranchi, r.falcone, michele.piunti}@istc.cnr.it

determined merging both expectations on food reward and presence (near the LOI). 2) Go to the identified food location and pick it up. After identifying a set of food items, agent add them to the belief base and head for the nearest one, observing topology constraints and bounds. 3) Transport Foods (one at a time) from the original location to the repository and deposit them. Agents use belief modules referring to paths (defined as lists of locations to pass through) in order to routinize crossing rooms and LOI. By releasing foods in the repository, they obtain a reward calculated on the basis of the original food reward decreased with a decay factor strictly depending on the transport time interval. Along the presented workflow, agents run up against fires or frozen puddles: in these cases agents present a general short term reaction: actions and speed are obligate to reduce, furthermore a greater amount of energy has to be spent.

Starting from a traditional approach to the BDI systems³, we developed an architecture dealing with 'the future', where intelligent agents build mental representations [4] for explicit expectations and risk avoidance. Stable 'personalities' have routinized intention-deliberation policies. To capture environment regularities (e.g. dangerous areas), watchful agents are enabled to choose those plans with the expected lower risk-value (and accordingly the higher safety) while rash agents prefer to choose the quickest and simplest ones. To deal with *adaptive* caution, an internal k-length buffer for registering events was constantly updated and monitored by adaptive agents: when the stored negative events exceed fixed threshold in a k-length history interval, they autonomously deliberate to increase the caution level. Adaptive agents shift among three kinds of more or less cautious behaviours (default, watchful and rash) adapting on the fly caution configurations to anticipate the dangerous risks of the crossing area. These caution competences can be reflected on the tuple of agent bounded resources (1), saying that an agent increasing caution level and becoming watchful has to reallocate resources: reducing speed s , increasing sensor rate Sr and increasing range of vision r . These adaptations heavily modify behaviour affecting the intentional stance and elicit trade-off in their performances. Note that relationships among different parameters characterizing these epistemic resources have to be analyzed with respect to the features of the different environments.

3 EXPERIMENTAL TEST

To measure agent effectiveness we considered environments with different levels of risk and evaluated agent energy in function of simulated time (fig. 1). **Experiment 1.** Safe environment (no Fires and

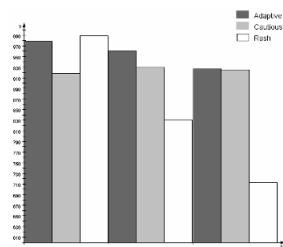


Figure 1. Adaptive, Cautious and Rash agent average energy comparison in safe, risky and unsafe environments.

no frozen puddles). We experience that to be cautious in safe worlds

³ System was built upon Jadex: see <http://vsis-www.informatik.uni-hamburg.de/projects/jadex/>.

is a drawback: best mean performances are those of the rash agent, because of his quickness and no resource consumption for useless and costly attention/caution capabilities (fig.1). **Experiment 2.** Risky environment (2 Fires and 2 frozen puddles). In the same time interval, adaptive agent outperforms the others: a stabilized cautious agent looks ahead and prevents accidents, while adaptive agent (with a modifiable caution) is able to minimize negative effects until its buffer reach the threshold to be watchful. On the contrary, rash agent is unable to react in safety mode because of its fixed low level of Caution. **Experiment 3.** Unsafe environment (4 Fires and 3 frozen puddles). It is interesting to note the different slopes of the functions (caused by the different energy consumption). Peaks indicate goal achievement while precipitous changes of slopes indicate unexpected accidents (agent near Fire or near a Frozen puddle) (fig. 2). Best energy performances come from adaptive agent that is able to optimize damages and resource consumption. Energy trend for the rash agents falls with deep peaks. Mean performances of cautious and adaptive agents are comparable (fig. 1): even if the adaptive agent continuously tries to modify its behaviours during the simulation (adapting it to the environment) at the end its global performances result comparable with the ones of the watchful agent that, in some sense, is "built" for well performing in a highly unsafe world.

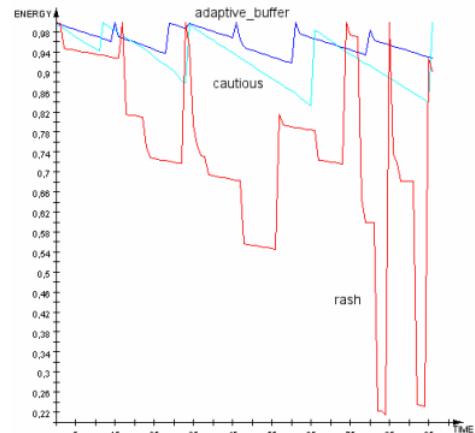


Figure 2. Agent comparison in unsafe environment shows best performance for adaptive agents.

REFERENCES

- [1] M. E. Bratman, D. Israel, and M. Pollack, 'Plans and resource-bounded practical reasoning', in *Philosophy and AI: Essays at the Interface*, The MIT Press, (1991).
- [2] C. Castelfranchi and R. Falcone, 'Principles of trust for mas: cognitive anatomy, social importance, and quantification', In Proc. of the Int. Conf. on Multi-Agent Systems (ICMAS'98), Paris, 72–79, (1998).
- [3] C. Castelfranchi and E. Lorini, 'Cognitive anatomy and functions of expectations', In proc. of IJCAI'03 workshop on cognitive modeling of agents and multi-agent interaction, Acapulco, Mexico, (2003).
- [4] E. Lorini and R. Falcone. Modeling expectations in cognitive agents. AAAI 2005 Fall Symposium: From Reactive to Anticipatory Cognitive Embodied Systems, 2005, Virginia.
- [5] L. Macedo and A. Cardoso, 'Exploration of unknown environments with motivational agents'. In 3rd Int. Conf. on Aut. Agents and MAS(AAMAS04), Columbia University, USA, (2004).
- [6] A.S. Rao and M.P. Georgeff, 'Bdi agents: From theory to practice', In Proc. of the 1st conf. on MAS (ICMAS95), (1995).
- [7] M. Schut and M. Wooldridge, 'Intention reconsideration in complex environments', in AGENTS '00: Proceedings of the fourth international conference on Autonomous agents, (2000).

AI and Music: Toward a Taxonomy of Problem Classes

Oliver Kramer¹ and Benno Stein² and Jürgen Wall³

Abstract. The application of Artificial Intelligence technology to the field of music has always been fascinating, from the first attempts in automating human problem solving behavior till this day. Human activities related to music vary in their complexity and in their amenability of becoming automated, and for both musicians and AI researchers various questions arise intuitively, e. g.: What are music-related activities or tasks that can be automated? How are they related to each other? Which problem solving methods have proven well? In which places does AI technology contribute?

Actually, the literature in the intersection of AI and music focuses on single problem classes and particular tasks only, and a comprehensive picture is not drawn. This paper, which outlines key ideas of our research in this field, provides a step toward closing this gap: it proposes a taxonomy of problem classes and tasks related to music, along with methods solving them.

Keywords AI and creativity, music, problem classes, taxonomy

1 INTRODUCTION

Music is an important part of almost every culture. Though music is an emotional thing and strongly connected to the human mind, a variety of tasks exists that—at least in part—can be automated, reaching from the analysis of acoustic data up to high level composition tasks like song arrangement. The identification and organization of such tasks within a taxonomy T of problem classes is of a high value:

- T may serve as a scheme for classifying existing problems and discovering unapparent similarities between them.
- Since T associates problems with methods solving them, a solution for a new but structurally similar task may be derived from existing solutions—where, at least, some proposition regarding its complexity can be stated.

Here we introduce such a taxonomy, from which Figure 1 captures the main aspects. The remainder of this section reports on existing classification work; Section 2 explains our classification paradigms, discusses music-related tasks under this perspective, and illustrates the research activities in the field of AI.

1.1 Existing Work

Roads identifies a set of general problem classes, covering the topics from composition to digital sound processing [7]. However, he neither defines relations between the problem classes nor unveils his classification paradigms. Similar shortcomings apply to the taxonomy of computer music by Pope [6]. Padadopoulos and Wiggins, as well as Tahiroglu, restrict their taxonomy to the classification of algorithmic composition systems along with the utilized computational

¹ International Graduate School of Dynamic Intelligent Systems. University of Paderborn, Germany. okramer@upb.de

² Faculty of Media / Media Systems. Bauhaus University Weimar, Germany. benno.stein@medien.uni-weimar.de

³ University of Paderborn, Germany. jwall@upb.de

methods [5, 8]. Ariza presents a survey of algorithmic composition tools and identifies seven primary descriptors [1].

Furthermore, schemes for special problem classes exist, such as the taxonomies for sound synthesis methods or for sequencer user interfaces by Duignan et al. [4]. Biles proposed a taxonomy for music and art systems that use evolutionary algorithms [2].

Our approach goes beyond existing work within two respects: it provides a framework that (i) is more generic, and (ii) makes the underlying classification paradigms explicit.

2 TAXONOMY OF PROBLEM CLASSES

The formation of our taxonomy is task-driven, i. e., the problem classes are formulated from a musician’s point of view. We identified three orthogonal paradigms that govern its structure and that proved to be qualified for classification purposes.

1. **Problem Type.** The most fundamental distinction follows Clancey’s ideas, where operations are grouped “[...] in terms of those that construct a system and those that interpret a system, corresponding to what is generally called synthesis and analysis.” [3]
2. **Modeling Level.** Defines the degree of abstraction; music becomes manifested both at a symbol level and a subsymbolic level. The transition between these levels is continuous and may form an additional level of sound:

Modeling level	Materialization	Human perception
symbolic	notes, accords	style, genre
sound	pitch, timbre	instrument, vocal tone
subsymbolic	amplitude, frequency	volume, sonority

3. **Arrangement Direction.** Explains music-related tasks as being of horizontal, vertical, or combined type:

Arrangement direction	Materialization	Musician’s viewpoint
horizontal	tone sequence	melody
combined	accord sequence	harmonizing
vertical	sound synthesis	instrumentation

Each task can be explained in the three dimensions. The composition of a melody theme, for example, is a horizontal, symbolic, synthesis task. By contrast, instrument identification is an analysis task that happens at the subsymbolic level.

2.1 Music Analysis

In the following, the analysis part of our taxonomy is presented, covering the fields of subsymbolic data analysis, i. e. the analysis of acoustic data, symbolic data analysis, and the connection between both, the so-called transcription.

Signal Analysis and Filter Theory Physical feature extraction pertains to both horizontal aspects along the time axis and vertical aspects like sound. Sound analysis basically exhibits vertical components, with the small exception of filter and amplifier envelopes

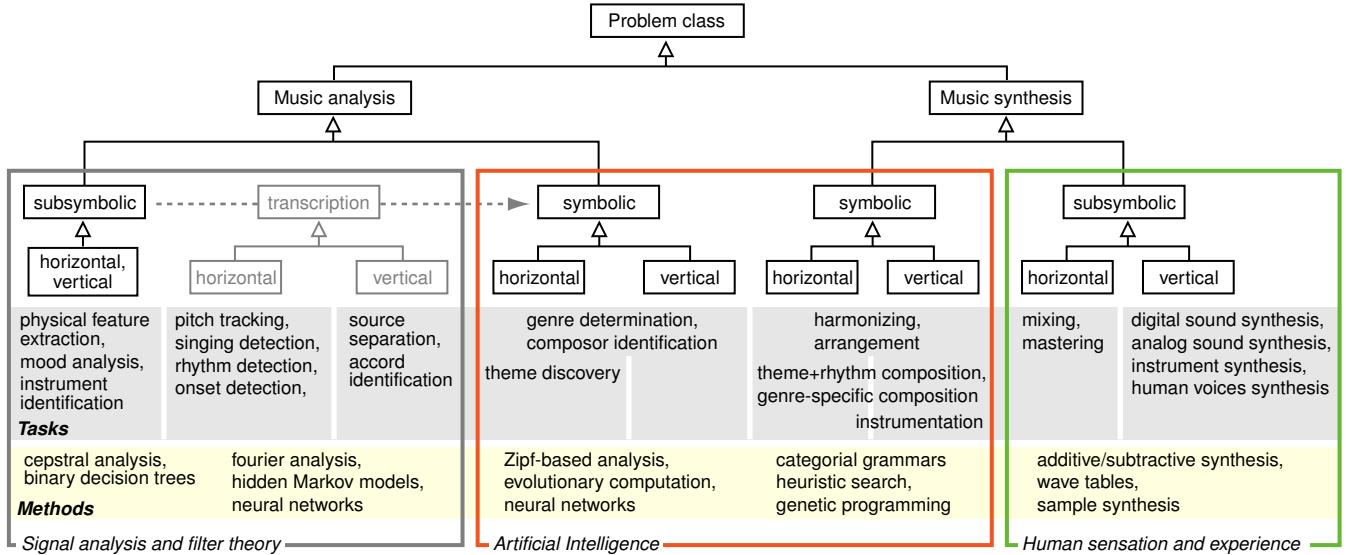


Figure 1. A taxonomy of problem classes in music. Note that the method section is not complete but shows representative methods.

or other parameter modulations. Further branches of subsymbolic music data analysis are mood and instrument classification. Transcription, which is the translation of acoustic information into notes, establishes an important field of operational tasks. It connects the subsymbolic with the symbolic classes as its methods work on acoustic data but make use of symbolic background knowledge. Except for source separation and accord detection the remaining tasks exhibit primarily horizontal dimensions.

Analysis by Artificial Intelligence The symbolic counterparts of both analysis and synthesis tasks are the challenging field for AI methods. A typical horizontal task is theme discovery, i.e., the identification of melodies and interludes. Other interesting problems are genre determination and note-based composer identification. Various AI methods are applied in this context, reaching from evolutionary methods to knowledge processing. Interestingly, the AI research activities in the synthesis field are prevalent compared to the activities in the analysis field (cf. Figure 2).

2.2 Music Synthesis

At the symbolic level the synthesis part of our taxonomy covers tasks from the field of song composition and arrangement. Here, AI methods demonstrate their power. The subsymbolic part of sound synthesis and instrumentation is only partly operational and comes within the limits of human sensation and experience.

Synthesis by Artificial Intelligence The automatic composition of songs is probably one of the most interesting AI applications. Again, we can distinguish between horizontal and vertical problems. Theme and rhythm composition is a genre-specific horizontal task. The tasks of harmonizing phrases and arranging a whole song exhibit both vertical and horizontal aspects, whereas the instrumentation task is rather vertical. The applied AI methods vary from categorical grammars to computational intelligence methods like genetic programming.

Human Sensation and Experience Horizontal aspects of the subsymbolic synthesis class concern mastering and mixing tasks; vertical aspects cover the task of sound synthesis. The latter ranges from natural sound synthesis of human voices and classical instruments to digital sound synthesis.

2.3 AI Research Activities

To get an overview of the research activities in the last forty years we have analyzed more than five hundred papers from the field of computer music. Figure 2 illustrates the activities for the symbolic problem classes. Observe that much more research took place in the synthesis branch compared to the analysis branch. Moreover, the distribution of the research activities for the different subproblems is quite interesting and may serve as a guide to future research.

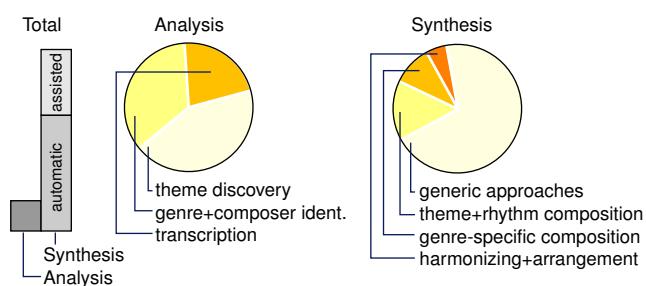


Figure 2. Distribution of research activities for AI and music. Basis for this investigation were 560 research papers, most of them published in the last twenty years. About 300 research papers fall into the area of symbolic problem classes (shown here), whereas 15% of them pertain to analysis tasks and 85% to synthesis tasks.

- [1] C. Ariza, ‘Navigating the Landscape of Computer-Aided Algorithmic Composition Systems’, *Proceedings of the International Computer Music Conference*, (2005).
- [2] J. A. Biles, ‘GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems.’, *Leonardo*, **36**, (2003).
- [3] W. J. Clancey, ‘Heuristic Classification’, *AI*, **27**, (1985).
- [4] M. Duignan, J. Noble, and R. Biddle, ‘A Taxonomy of Sequencer User-Interfaces’, *Proceedings of the International Computer Music Conference*, (2005).
- [5] G. Papadopoulos and G. Wiggins. AI Methods for Algorithmic Composition: A Survey , a Critical View and Future Prospects.
- [6] S. T. Pope, ‘A Taxonomy of Computer Music’, *Contemporary Music Review: Live Electronics*, **13**, (1995).
- [7] C. Roads, ‘Research in music and artificial intelligence’, *CM Computing Surveys*, **17**, (1985).
- [8] K. Tahiroglou, *From Me to US, a computer generated music installation*, Univ. of Art and Design, Helsinki, 2003.

Using Emotions for Behaviour-Selection Learning

Maria Malfaz ¹ and Miguel A. Salichs ²

Abstract. Emotions play a very important role in human behaviour and social interaction. In this paper we present a control architecture which uses emotions in the behaviour selection process of autonomous and social agents. The state of the agent is determined by its internal state, defined by its dominant motivation, and its relation with the external objects including other agents. The behaviour selection is learned by the agent using standard and multiagent Q-learning algorithms. The considered emotions are fear, happiness and sadness. The role of these emotions in this architecture is different, while the learning algorithms use happiness/sadness of the agent as positive/negative reinforcement signals, the emotion fear is used to prevent the agent of choosing dangerous actions as well as a motivation.

1 INTRODUCTION

In previous works [7] an emotion-based architecture was proposed for an autonomous and social robot. The use of emotions in this architecture is oriented to the behavior selection process rather than the human-robot interaction issue. Several authors such as Canamero [3] and Gadanho [5] have used emotions to improve the adaptation of the robot to the environment and therefore the autonomy of the robot. Many others such as Breazeal [2], Fujita [4], Shibata et al [11] have implemented emotional models on robots to enhance the human-robot interaction. Their robots, Kismet and Leonardo, AIBO and Necoro, include the possibility of showing emotions, by facial and sometimes body expressions. We intend to make use of emotions in robots trying to imitate their purpose in nature, which includes, but is not limited to, interaction.

We think that the role that each emotion plays, and how the mechanisms associated to each one work are very specific. That means that each emotion must be incorporated to the robot in a particular way. In a recent paper [10] we have shown how some emotions such as happiness, sadness and fear are used to learn the right behaviour policy for each situation.

2 CONTROL ARCHITECTURE

The control architecture proposed here, see Fig. 1, has been designed for an autonomous robot. An autonomous agent/robot should be capable of determining its goals, and it must be capable of selecting the most suitable behaviour in order to reach them. Similarly to other authors [3], [2], [12], our agents autonomy relies on a motivational model.

¹ Carlos III University of Madrid, Spain, email: mmalfaz@ing.uc3m.es

² Carlos III University of Madrid, Spain, email: salichs@ing.uc3m.es

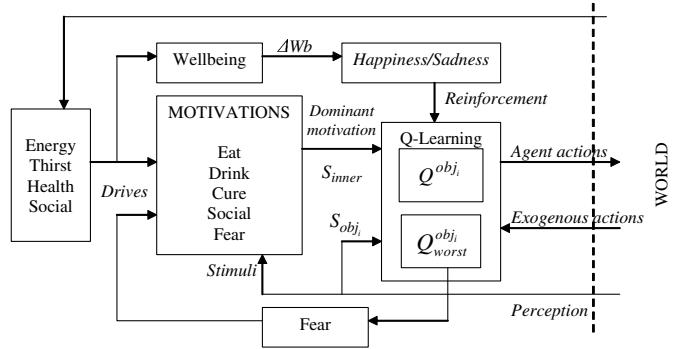


Figure 1. Control Architecture

2.1 Drives and Motivations

Motivations can be seen as homeostatic processes, which maintain a controlled physiological variable within a certain range. Homeostasis means maintaining a stable internal state [1]. The drives constitute urges to action based on bodily needs related to self-sufficiency and survival.

In order to model motivations, the hydraulic model of motivation described in [6] has been used as an inspiration. In this model, internal drive strength interacts with external stimulus strength. The intensity of motivations is a combination of the intensity of the related drive and the presence of the related external stimuli. The external stimuli are the different objects and agents that the player can find in the virtual world during the game. According to our model a motivation can obtain a high intensity due to two reasons: 1) the correspondent drive is the highest or 2) The correct stimulus is present. The dominant motivation will be the one with the highest intensity and the possibility of no dominant motivation exists.

2.2 Wellbeing

As it is shown in Fig. 1 the wellbeing of the agent is a function of the values of the drives. In [8] the wellbeing is defined in such a way that it is maximum when all the needs (drives) of the agent are satisfied. There exist some personality factors that weight the importance of the values of the drives on the wellbeing of the agent.

On the other hand, based on the definitions of the emotions given in [7], the emotions of happiness and sadness depends on the variation of the wellbeing of the agent. When the wellbeing of the agent is increased a certain amount, happiness arises, on the contrary, sadness is produced when the wellbeing decreases a certain amount.

2.3 Behaviour Selection

In this architecture the agent learns, using different reinforcement learning algorithms (RL), the best behaviour at each step using the emotions of happiness/sadness as the reward. This use of emotions as reinforcement has been proposed previously by Rolls [9]. Therefore, in this architecture behaviours are not selected to satisfy the goals determined by the dominant motivation but to optimize the wellbeing of the agent. This implies that the final goal of the agent is to reach Happiness and avoid Sadness.

In the proposed architecture the agent will use the Q-learning algorithm as the RL algorithm when the agent is not interacting with other players. In the case of social interaction, the agent uses a Q-multiagent RL algorithm.

The state of the agent is the aggregation of his inner state, the dominant motivation S_{inner} , and the states S_{obj} related to each of the external objects, including external agents, which can interact with him. For the RL algorithms the objects are considered as independent. This means that the state of the agent in relation with each object is $S \in S_{inner} \times S_{obj_1}$.

2.4 Fear

Fear is produced when the agent knows that something bad may happen. In this architecture there are two types of fear, one related to actions executed by the agent and the other related to exogenous actions carried out by other elements of the environment such as other agents.

2.4.1 To be afraid of executing risky actions

In order to avoid risky actions, the worst result experimented by the agent for each pair action-state is stored in a variable called $Q_{worst}^{obj_i}$. The effect of being afraid can be considered by following a policy that considers not only the average of the result obtained but also the worst one. We define a daring factor that measures the daring degree of the agent. By varying this parameter the agent gives more or less importance to the worst result obtained making the agent fearful or fearless respectively.

2.4.2 To be afraid of risky exogenous actions

When the agent may suffer some negative effects in a state as a consequence of exogenous events, feels fear. "Fear" is expressed as a drive D_{fear} . The fear drive is equally treated as the rest of drives, and its related motivation could be the dominant one. In this case, the agent will learn by itself what to do when it is afraid.

3 EXPERIMENTS

This control architecture has been tested on virtual MUD players (a text-based multi user game), who "live" in a virtual world. This game gave us the possibility of creating different 2-D environments to play in. This environment is fully described in [8].

The results obtained in [8] showed that using happiness and sadness as the reinforcement signal of the RL algorithms the agent learns the correct policy in order to survive in his virtual world. Moreover, the quality of life of the agent is quite good following the learned policy.

In recent experiments it has been proved that following a near fearless policy that consider the optimal policy as well as the worst

expected results in some degree, improves the performance of the agent.

4 CONCLUSIONS AND FUTURE WORKS

This control architecture implements emotions on the learning behaviour process of an artificial agent, obtaining quite good results in relation with the overall performance. The learning algorithms use happiness/sadness of the agent as positive/negative reinforcement signals. Fear is used to prevent the agent choosing dangerous actions or being in dangerous states where non-controlled exogenous events, produced by external objects or other agents, could danger him.

Using this control architecture the agent learns behavioural patterns, this means that the agent learns the sequence of behaviours to be executed in order to satisfy the drive related to the dominant motivation. The agent learns, for example, that he has to get food when he is hungry, even when this behaviour is was not previously linked with the "Eat" motivation.

The final goal of the project is to implement this architecture on a real personal robot. Meanwhile in the next future, another emotion is going to be implemented: Anger. This emotion will be produced when another active object, such as other agent, could reduce the expectancies of the wellbeing of the agent.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the funds provided by the Spanish Government through the projects named "Personal Robotic Assistant" (PRA) and "Peer to Peer Robot-Human Interaction" (R2H), of MEC (Ministry of Science and Education).

REFERENCES

- [1] Kent C. Berridge., 'Motivation concepts in behav-ioural neuroscience', *Physiology and Behaviour*, (81), 179–209, (2004).
- [2] C. Breazeal, *Designing Sociable Robots*, The MIT Press, 2002.
- [3] L. Caamero, 'Modeling motivations and emotions as a basis for intelligent behavior', in *First International Symposium on Autonomous Agents (Agents'97)*, 148-155. New York, NY: The ACM Press., (1997).
- [4] Masahiro Fujita, 'Aibo: Toward the era of digital creatures', *The International Journal of Robotics Research*, (vol 20, n 10), 781–794, (October 2001).
- [5] Sandra Gadinho, *Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions*, Ph.D. dissertation, University of Edinburgh, 1999.
- [6] K. Lorentz and P. Leyhausen, *Motivation of human and animal behaviour; an ethological view*, volume xix, New York: Van Nostrand-Reinhold, 1973.
- [7] M. Malfaz and M.A. Salichs, 'A new architecture for autonomous robots based on emotions', in *Fifth IFAC Symposium on Intelligent Autonomous Vehicles. Lisbon, Portugal*, (July 2004).
- [8] M. Malfaz and M.A. Salichs, 'Learning behaviour-selection algorithms for autonomous social agents living in a role-playing game', in *AISB'06: Adaptation in Artificial and Biological Systems. University of Bristol, Bristol, England*, (April 2006).
- [9] Edmund Rolls, *Emotions in Humans and Artifacts*, chapter Theory of emotion, its functions, and its adaptive value, MIT Press, 2003.
- [10] M.A. Salichs and M. Malfaz, 'Using emotions on autonomous agents. the role of happiness, sadness and fear', in *AISB'06: Adaptation in Artificial and Biological Systems. University of Bristol, Bristol, England*, (2006).
- [11] Tanie K. Shibata T, Tashima T, 'Emergence of emotional behaviour through physical interaction between human and robot', in *the 1999 IEEE International Conference on Robotics and Automation (ICRA'99)*, (September 1999).
- [12] J. Velasquez, 'An emotion-based approach to robotics', in *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (1999).

2. Constraints and Search

This page intentionally left blank

Efficient Handling of Complex Local Problems in Distributed Constraint Optimization ¹

David A. Burke and Kenneth N. Brown²

1 INTRODUCTION

Many distributed constraint optimisation algorithms require each agent to have a single variable. For agents with multiple variables, a standard approach is to compile the local problem down to a new variable whose domain is the set of all local solutions. We present two modifications to this method, which (i) reduce problem size by removing interchangeable and dominated local solutions, and (ii) speed up search by identifying values that are interchangeable with respect to specific agents. We show that the modifications give orders of magnitude improvement over the basic compilation.

2 BACKGROUND

A Distributed Constraint Optimisation Problem consists of a set of *agents*, $A = \{a_1, a_2, \dots, a_n\}$, and for each agent a_i , a set $X_i = \{x_{i1}, x_{i2}, \dots, x_{im_i}\}$ of *variables* it controls, such that $\forall i \neq j \ X_i \cap X_j = \emptyset$. Each variable x_{ij} has a corresponding domain D_{ij} . $X = \bigcup X_i$ is the set of all variables in the problem. $C = \{c_1, c_2, \dots, c_t\}$ is a set of *constraints*. Each c_k has a *scope* $s(c_k) \subseteq X$, and is a function $c_k : \prod_{ij: x_{ij} \in s(c_k)} D_{ij} \rightarrow \mathbb{N}$. The *agent scope*, $a(c_k)$, of c_k is the set of agents that c_k acts upon: $a(c_k) = \{a_i : X_i \cap s(c_k) \neq \emptyset\}$. An agent a_i is a *neighbour* of an agent a_j if $\exists c_k : a_i, a_j \in a(c_k)$. A *global assignment*, g , is the selection of one value for each variable in the problem: $g \in \prod_{ij} D_{ij}$. A *local assignment*, l_i , to an agent a_i , is an element of $\prod_j D_{ij}$. Let t be any assignment, and let Y be a set of variables, then $t|_Y$ is the projection of t over the variables in Y . The global objective function, F , assigns a cost to each global assignment: $F : \prod_{ij} D_{ij} \rightarrow \mathbb{N} :: g \mapsto \sum_k c_k(g|_{s(c_k)})$. An optimal solution is one which minimises F . The solution process, however, is restricted: each agent is responsible for the assignment of its own variables, and thus agents must communicate with each other, describing assignments and costs, in order to find a globally optimal solution.

Most DCOP algorithms assume that each agent controls only a single variable. This assumption is justified by two standard reformulations [2], by which any DCOP problem with complex local problems (i.e. multiple variables in each agent) can be transformed to give exactly one variable per agent: (i) *Compilation*: for each agent, define a variable whose

domain is the set of solutions to the original local problem; (ii) *Decomposition*: for each variable in each local problem, create a unique agent to manage it.

3 IMPROVING COMPIILATION

To apply the *basic compilation* method to a DCOP: (i) for each agent a_i , create a new variable z_i , whose domain $D_i = \prod_j D_{ij}$ is the set of all solutions to the agent's internal problem; (ii) for each agent a_i , add a unary constraint function f_i , where $\forall l \in D_i, f_i(l) = \sum_{j: s(c_j) \subseteq X_i} c_j(l|_{s(c_j)})$ (i.e. the cost is the sum of the costs from all constraints which act on a_i only); (iii) for each set of agents $A_j = \{a_{j1}, a_{j2}, \dots, a_{jp_j}\}$, let $R_j = \{c : a(c) = A_j\}$ be the set of constraints whose agent scope is A_j , and for each $R_j \neq \emptyset$, define a new constraint $C_j : D_{j1} \times D_{j2} \times \dots \times D_{jp_j} \rightarrow \mathbb{N} :: l \mapsto \sum_{c \in R_j} c(l|_{s(c)})$, equal to the sum of the constraints in R_j (i.e. construct constraints between the agents' new variables, that are defined by referring back to the original variables in the problem). In an optimisation problem, every set of assignments of values to variables is a valid solution giving a domain of size $|D_i| = \prod_j |D_{ij}|$ for each agent a_i . The solution space for the problem is $\prod_{i=1}^n |D_i|$.

For an agent with a complex local problem, only external variables (those that have constraints to other agents) can have a direct impact on other agents. We produce an *improved compilation* by recognising that any local solutions that have identical assignments to those external variables are equivalent with respect to the distributed problem. If there is more than one optimal local solution with the same assignments to external variables, the solutions are *fully interchangeable*, and so only one is required. Also, solutions with identically assigned external variables but with sub-optimally assigned other variables are *strictly dominated* and can be ignored. We refine the compilation so that we only find *one* optimal local solution for each combination of external variables. For each agent a_i , let $p_i = \{x_{ij} : \forall c \ x_{ij} \in s(c) \rightarrow s(c) \subseteq X_i\}$ be its *private* variables – variables which are not directly constrained by other agents' variables – and let $e_i = X_i \setminus p_i$ be its *external* variables – variables that do have direct constraints with other agents. For each a_i , create a new variable z'_i with domain $D'_i = \prod_{j: x_{ij} \in e_i} D_{ij}$, and add a function f'_i , where $\forall l \in D'_i, f'_i(l) = \min\{f_i(t) : t \in D_i, t|_{e_i} = l\}$. That is, D'_i contains all assignments to the external variables, and their cost is the minimum cost obtained when they are extended to a full local assignment for a_i . The new constraints are defined as in the basic compilation (for each $R_j \neq \emptyset$,

¹ This work is supported by SFI under Grant No. 03/CE3/I405. We thank the developers of ADOPT for making their code available, and BCRI at UCC for their computing resources.

² Centre for Telecommunications Value-chain Research and Cork Constraint Computation Centre, Dept. Comp. Sc., UCC, Ireland

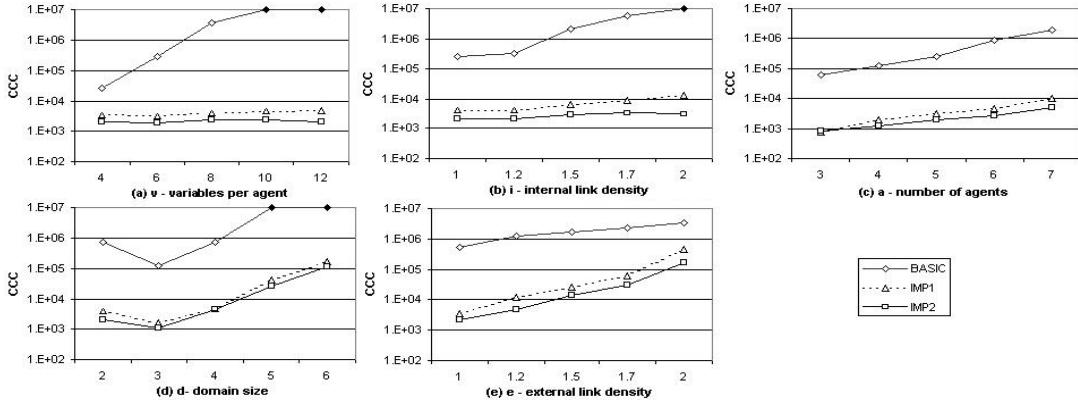


Figure 1. Concurrent Constraint Checks (cutoff = 10,000,000).

$C'_j : D'_{j1} \times D'_{j2} \times \dots \times D'_{jp_j} \rightarrow \mathbb{N} :: l \mapsto \sum_{c \in R_j} c(l_{\downarrow s(c)}))$, but will act on smaller sets of tuples. In our reduced compilation, the size of a domain is $|D'_i| = \prod_{j: x_{ij} \in e_i} |D_{ij}|$, which is a reduction over the basic compilation by a factor of $\prod_{j: x_{ij} \in p_i} |D_{ij}|$.

4 INTERCHANGEABILITY IN DCOP

We introduce the concept of sub-neighbourhood interchangeability and apply it to distributed search using compiled values. For each agent a_i and for a set of agents S , let $h_i^S = \{x_i : \exists c : x_{ij} \in S(c) \wedge a(c) \cap S \neq \emptyset\}$ be the set of original variables of a_i that are adjacent to original variables of the agents in S . Two compiled values, $l_a, l_b \in D_i$ are *sub-neighbourhood interchangeable* (SNI) with respect to the agents in S ($l_a \equiv_i^S l_b$), if both values represent identical assignments to the variables of h_i^S : $l_a \equiv_i^S l_b \leftrightarrow l_a \upharpoonright h_i^S = l_b \upharpoonright h_i^S$.

We apply SNI sets to the ADOPT algorithm [1]. In ADOPT, agents are prioritised into a tree. Agents send their values to all neighbours lower in the priority tree, and receive costs only from direct children. Considering each subtree separately, let S be the set of lower priority neighbours of a_i , lying in the subtree rooted by a_s . For each subtree, partition a_i 's values into SNI sets, such that $\Phi_i^S(x)$ is a function returning the SNI set to which x belongs: $\forall l_a, l_b \in D_i, l_a \equiv_i^S l_b \leftrightarrow \Phi_i^S(l_a) = \Phi_i^S(l_b)$. Then, ADOPT can be modified such that if a_i receives a cost ε from a_s with a compatible context, the costs of all values interchangeable with the current value x are updated: $\forall l \in \Phi_i^S(x), \text{cost}(l, a_s) \leftarrow \varepsilon$.

5 EXPERIMENTS

We compare the effect of the two modifications to the basic compilation on random binary graph colouring problems. The problems are characterised by a 5-tuple $\langle a, e, v, i, d \rangle$: a is the number of agents, e is the external link density (# inter-agent constraints = ea), v is the number of variables per agent, i is the internal link density (# intra-agent constraints = iv) and d is the domain size of each variable. Our base problem setting is $\langle 5, 1, 6, 1, 2 \rangle$. We then compare the algorithms varying v, i, a, d and e in turn, averaging over 20 test instances. Both the basic and reduced compilations are generated using Ilog JSolver. We implement the SNI sets in ADOPT and run it in

a simulated distributed environment: we use one machine but each agent runs asynchronously on its own thread.

The time required to compile agents into a single variable is always faster in the improved compilation – up to 2 orders of magnitude faster for some parameter settings. In the distributed search, we use the Concurrent Constraint Checks metric to measure performance. In Figure 1, plotted on a log scale, we compare the original compilation method *BASIC*, with our improved method *IMP1* and also *IMP2*, which uses both the improved compilation and SNI sets. The reduced domains resulting from our improved compilation, *IMP1*, give orders of magnitude improvements over the basic compilation for most parameter settings. In particular, as the number of variables in the local problem increases, the impact on our improved reformulation is minimal, while the basic reformulation quickly becomes very difficult to compute. Tests varying the internal link density, number of agents and domain size all demonstrate improvements of at least 2 orders of magnitude over *BASIC* (runs of *BASIC* that exceeded 10^7 CCC were cut off, and are shown shaded). *IMP2* also outperforms *IMP1* by on average 45%: using the SNI sets that occur in the compilation speeds up the distributed search.

6 CONCLUSION

A standard technique for handling complex local problems in DCOP is to convert multiple local variables to a single variable whose domain is the set of all local solutions. We have presented two advances on this method: we represent only one optimal local solution for each combination of external variables, discarding interchangeable and dominated solutions; and we identify values that are interchangeable with respect to neighbouring agents. We have evaluated the methods using the ADOPT algorithm, and they give orders of magnitude improvements over the basic compilation.

REFERENCES

- [1] P. Modi, W. Shen, M. Tambe, and M. Yokoo, ‘Adopt: Asynchronous distributed constraint optimization with quality guarantees’, *Artificial Intelligence*, **161**(1–2), 149–180, (2005).
- [2] M. Yokoo and K. Hirayama, ‘Algorithms for distributed constraint satisfaction: A review’, *Autonomous Agents and Multi-Agent Systems*, **3**(2), 185–207, (2000).

Visualization for Analyzing Trajectory-Based Metaheuristic Search Algorithms

Steven HALIM and Roland H.C. YAP¹ and Hoong Chuin LAU²

1 INTRODUCTION

Most combinatorial (optimization) problems are computationally intractable. We often have to be satisfied with *good* solutions and typically metaheuristic algorithms (such as various forms of local search, tabu search, etc) are used. Given the heuristic nature of such search algorithms, there are two important considerations in designing a metaheuristic algorithm:

- Choice of metaheuristics to employ, which may include problem specific tweaks;
- Selecting the appropriate parameters to drive the heuristics.

We call this problem of designing the appropriate metaheuristic problem for a combinatorial (optimization) problem, the *metaheuristic tuning problem* [1, 3, 7]. Anecdotal evidence suggests that tuning takes a major effort, i.e. [1] states that 90% of the design and testing time can be spent fine-tuning the algorithm.

Although it can be easy to come up with a variety of metaheuristics, tuning the metaheuristic implementation is not straightforward. Firstly, the metaheuristics may not be well understood. It might also be applied to problems which may not have been studied. Thus, it may not be clear how to perform tuning. Secondly, the space in which the tuning can operate on is very rich — there are many ways of combining different kinds of metaheuristics each with their own choice of strategies and variations. Furthermore, they each have their own parameters. In this paper, we take a broad view of the metaheuristic tuning problem and understand it to also encompass algorithm design and debugging.

Traditionally, the approach for to the tuning problem is either manual experimentation or more automated approaches such as finding the best parameter values [1], best configuration [3], or self-tuning algorithms [2]. In this paper, we take a different approach which takes a human/programmer perspective — how to aid human in solving the tuning problem. Like human-guided search [6], we believe that a cooperative paradigm with the human in the loop can be productive. The difference with human-guided search is that it is concerned with using the human to produce better solutions, while we want to use the human to produce better metaheuristic algorithms.

Ultimately, we would like a man-machine cooperation which can help the human to debug, analyze and improve a metaheuristic algorithm for particular problems. Some of the questions which we would like to help answer are:

1. Does the actual search behavior match how we think the algorithm should behave?

2. Are there signs of cycling behavior?
3. How does the metaheuristic algorithm make progress?
4. How effective the metaheuristic in conducting intensification and/or diversification?
5. How wide is the search coverage?
6. How far is the (greedy) initial solution to the best found solution?
7. Does the search quickly identify the region where the best solutions found reside or does it wander elsewhere?
8. How do the trajectories of two different metaheuristics compare?
9. What is the effect of modifying certain parameters, components or strategies with respect to the search behavior?

In this paper, we focus on the tuning problem for metaheuristic algorithms which are search trajectory based, such as iterated local search, simulated annealing, tabu search, etc. We believe that a good approach to get man-machine cooperation is with an interactive visualisation of the search trajectory. One way of understanding how a program works is with a debugger. We have built the analog of a debugger, the visualizer VIZ, for understanding search trajectories of metaheuristic algorithms. VIZ provides visualization and animation (forwards and backwards) in time. It has multiple visualizations: (i) problem independent visualization which allows it to be used on a broad range of metaheuristic algorithms in a generic way; and (ii) it can also make use of problem specific visualizations. Although VIZ is still in prototype stage, we believe that it is the first serious attempt at an interactive tool with emphasis on the human computer interaction aspects to help humans understand the dynamic behavior of metaheuristic algorithms and guide the tuning process.

2 SEARCH TRAJECTORY VISUALIZATION

Visualizing the search trajectory, i.e. local movement of the current solution along the search space is difficult because the problem is usually in very high dimensions and the search space is also extremely large. We are only aware of (very) few proposals for search trajectory visualization.

N-to-2 space mapping [4] gives a mapping from a higher dimensional problem space to 2-D for visualizing, e.g. coverage of search space. However, the proposed visualization is crowded and static.

In our earlier work, V-MDF [7], we proposed a visualization called the *distance radar*. A current set of elite solutions is chosen, called *anchor points*. The distance radar displays two graphs: the distance between the current solution in the search trajectory w.r.t the set of anchor points (one sorted by fitness and the other by recency). While V-MDF can help answering *some* of the questions about how the search trajectory is behaving, e.g. questions 1/2/4/7, the visualization is not very intuitive for answering questions 3/5/6/8/9. Another drawback is that the graphs can change simply because the elite set

¹ National University of Singapore, {stevenha,ryap}@comp.nus.edu.sg

² Singapore Management University, hclau@smu.edu.sg

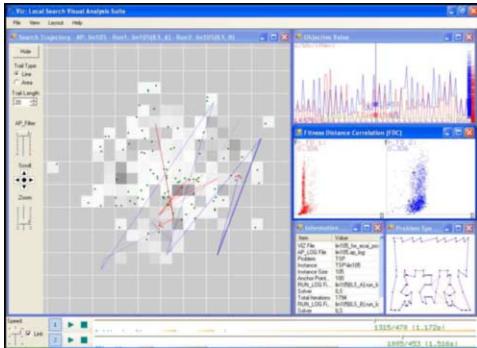


Figure 1. Screen shot of VIZ with multiple visualizations

changes with time. The visualization is less effective because the visualization is essentially one dimensional, graphs which show distance information.

With VIZ, we want to ensure that the visualization can be intuitive and exploit the fact that humans are good at recognizing visual patterns, in particular, not just in a static image but also how things change which exploits movement and temporal features. We make use of an abstract 2-D visualization where instead of trying to map the anchor points and points along the search trajectory from a high dimension to 2-D, we consider abstract points which are differentiated from each other using a distance metric. We do not have the space to discuss the visualization in detail — an example of a possible metric is the Hamming distance. These points can then be laid out in 2-D to approximate a visualization of the abstract points, we have used a spring model for the layout [5]. The search trajectory is then drawn interactively as a trail as shown in Fig. 1 and 2. The strength of this approach is that we now have a problem independent visualization which can be used with a suitably defined metric to demonstrate search trajectories.

3 THE VISUALIZER: VIZ

Fig. 1 shows VIZ's GUI. VIZ functions in interactive fashion as a kind of video player to play back an animation of the search trajectory drawn as a trail. The trail fades with time so that the animation does not clutter up the screen. Colors are used to compare two metaheuristic algorithms. Anchor points³ are landmarks to indicate the progress of the search trajectory in the abstract search space. Auxiliary visualizations are used to complement search trajectory visualization, e.g. time series of objective values, problem specific, etc. The animation of the geometric pattern of the trail, its relation to the anchor points, and auxiliary visualizations can be used to answer *all* the questions posed in Sec. 1.

Space doesn't permit more details, rather we use the following example which demonstrates how one can visualize the differences between two variants of Iterated Local Search (ILS) on TSP [9]. In TSP, it is conjectured that a heuristic algorithm should exploit the “Big Valley” property, a region in the TSP search space where most local optima (including the global optima) lie [8].

In this example, we want to know whether our algorithms make use of this property. We created two variants of the ILS algorithm which are run on the same TSP instance: ILS_A and ILS_B . The visualization of the search trajectories from ILS_A and ILS_B is shown in Fig. 1 (as a trail) and Fig. 2 (as region coverage). At a glance,

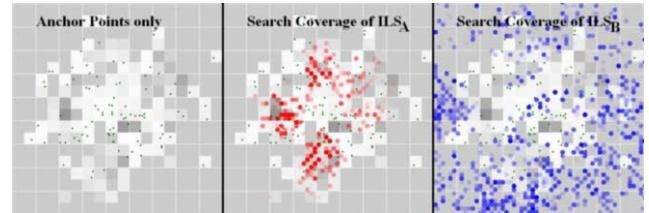


Figure 2. Search trajectory of ILS_A vs ILS_B on the same TSP instance

one can check the existence of the search intensification indicative of searching in a “Big Valley” by checking whether the search trajectory covers region with a cluster of many good anchor points (TSP local optima). Fig. 2 shows that the search trajectory for ILS_A is concentrated in the middle of the screen (indicative of a “Big Valley” region). On the other hand, after similar number of iterations, the coverage of ILS_B seems to be more diverse. ILS_B seems to spend most of its time exploring areas far from the cluster of good anchor points.

This gives a possible explanation of the algorithm behavior and suggests directions for tuning our algorithms. If our solution behaves like ILS_A , we know that we are on the right track, perhaps only few minor adjustments are needed. On the other hand, if it behaves like ILS_B , we may want to modify our ILS algorithm such that it is more focused.

4 CONCLUSION

We have presented a new approach for visualizing search trajectory and introduced the visualizer tool VIZ. This is not intended to replace the existing analysis tools, but rather it is meant to augment the existing tools to help the algorithm designer better understand the behavior of a trajectory based metaheuristic search algorithm and to debug and to tune the algorithm. A prototype of VIZ which is under continuous development is at: <http://www.comp.nus.edu.sg/~stevenha/viz>

ACKNOWLEDGEMENTS

This paper was written while Roland Yap was visiting the Swedish Institute of Computer Science and their support and hospitality are gratefully acknowledged.

REFERENCES

- [1] B. Adenso-Díaz and M. Laguna, ‘Fine-tuning of Algorithms Using Fractional Experimental Designs and Local Search’, *Operations Research*, (2005).
- [2] R. Battini and G. Tecchioli, ‘The Reactive Tabu Search’, *ORSA Journal of Computing*, **6**(2), 126–140, (1994).
- [3] M. Birattari, *The Problem of Tuning Metaheuristics as seen from a machine learning perspective*, Ph.D., U. Libre de Bruxelles, 2004.
- [4] M. Kadluczka, P.C. Nelson, and T.M. Tirpak, ‘N-to-2 Space Mapping for Visualization of Search Algorithm Performance’, in *ICTAI*, 508–513, 2004.
- [5] T. Kamada and S. Kawai, ‘An algorithm for drawing general undirected graphs’, *Information Processing Letters*, **31**(1), 7–15, (1989).
- [6] G.W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher, ‘Human-Guided Tabu Search’, in *AAAI*, 41–47, 2002.
- [7] H.C. Lau, W.C. Wan, and S. Halim, ‘Tuning Tabu Search Strategies via Visual Diagnosis’, in *MIC*, 2005.
- [8] P. Merz, *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*, Ph.D., U. of Siegen, 2000.
- [9] T. Stuetzle and H. Hoos, ‘Analyzing the Run-Time Behavior of Iterated Local Search for the TSP’, in *MIC*, 1999.

³ These are slightly different from V-MDF, as anchor points in VIZ are now static: carefully chosen from the log files to achieve diversity and quality. The fitness of the anchor points is indicated via the contour map.

Bipolar preference problems

Stefano Bistarelli*, Maria Silvia Pini**, Francesca Rossi** and K. Brent Venable**¹

1 INTRODUCTION

Real-life problems present several kinds of preferences. In this paper we focus on problems with both positive and negative preferences, that we call *bipolar problems*. Although seemingly specular notions, these two kinds of preferences should be dealt differently to obtain the desired natural behaviour. In fact, assume, for example, to have a scenario with two objects A and B. If we like both A and B, i.e., if we give to A and B positive preferences, then the overall scenario should be more preferred than having just A or B alone, and so the combination of such a preferences should give an higher positive preference. Instead, if we dislike both A and B, i.e., if we give to A and B negative preferences, then the overall scenario should be less preferred than having just A or B alone and so the combination of such a negative preferences should give a lower negative preference. When dealing with both kinds of preferences, it is natural to express also indifference, which means that we express neither a positive nor a negative preference over an object. A desired behaviour of indifference is that, when combined with any preference, it should not influence the overall preference.

Finally, besides combining positive preferences among themselves, and also negative preferences among themselves, we also want to be able to combine positive with negative preferences, allowing compensation. For example, if we have a meal with meat (which we like very much) and wine (which we don't like), then what should be the preference of the meal? To know that, we should be able to compensate the positive preference given to meat with the negative preference given to wine.

In this paper we start from the soft constraint formalism [2] based on c-semirings, that models only negative preferences. We then extend it via a new mathematical structure, which allows to handle positive preferences as well and we address the issue of the compensation between positive and negative preferences, studying the properties of this operation. Parts of this paper have appeared in [3].

2 SOFT CONSTRAINT FORMALISM

A soft constraint [2] is a classical constraint [4] where each instantiation of its variables has an associated value from a (totally or partially ordered) set. This set has two operations, which makes it similar to a semiring, and is called a c-semiring. A c-semiring is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ where: A is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$ is commutative, associative, idempotent, $\mathbf{0}$ is its unit element, and $\mathbf{1}$ is its absorbing element; \times is associative, commutative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. Consider the relation \leq_S over A such that $a \leq_S b$ iff $a + b = b$. Then: \leq_S is a partial order; $+$

¹* : University of Pescara and CNR, Pisa, Italy. E-mail: bista@sci.unich.it and Stefano.Bistarelli@iit.cnr.it. ** : Department of Pure and Applied Mathematics, University of Padova, Italy. E-mail: {mpini,frossi,kvenable}@math.unipd.it.

and \times are monotone on \leq_S ; $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; (A, \leq_S) is a lattice and, $\forall a, b \in A, a + b = lub(a, b)$. Moreover, if \times is idempotent, then (A, \leq_S) is a distributive lattice and \times is its glb. Informally, the relation \leq_S gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that b is better than a .

Given a c-semiring $S = (A, +, \times, \mathbf{0}, \mathbf{1})$, a finite set D (the domain of the variables), and an ordered set of variables V , a constraint is a pair $\langle def, con \rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow A$. Therefore, a constraint specifies a set of variables (the ones in con), and assigns to each tuple of values of D of these variables an element of A . A soft constraint satisfaction problem (SCSP) is just a set of soft constraints over a set of variables. For example, fuzzy CSPs [5] and weighted CSPs [2] are SCSPs that can be modeled by choosing resp. c-semirings $S_{FCSP} = ([0, 1], max, min, 0, 1)$ and $S_{WCSP} = (\mathbb{R}^+, min, sum, +\infty, 0)$.

3 NEGATIVE PREFERENCES

The structure we use to model negative preferences is exactly a c-semiring [2] as described in the previous section. In fact, in a c-semiring there is an element which acts as indifference, that is $\mathbf{1}$, since $\forall a \in A, a \times \mathbf{1} = a$, and the combination between negative preferences goes down in the ordering (in fact, $a \times b \leq a, b$), that is a desired property. This interpretation is very natural when considering, for example, the weighted c-semiring $(\mathbb{R}^+, min, +, +\infty, 0)$. In fact, in this case the real numbers are costs and thus negative preferences. The sum of different costs is worse in general w.r.t. the ordering induced by the additive operator (that is, min) of the c-semiring. From now on, we will use a standard c-semiring to model negative preferences, denoted as: $(N, +_n, \times_n, \perp_n, \top_n)$.

4 POSITIVE PREFERENCES

When dealing with positive preferences, we want two main properties to hold: combination should bring to better preferences, and indifference should be lower than all the other positive preferences. A positive preference structure is a tuple $(P, +_p, \times_p, \perp_p, \top_p)$ s.t. P is a set and $\top_p, \perp_p \in P$; $+_p$, the additive operator, is commutative, associative, idempotent, with \perp_p as its unit element ($\forall a \in P, a +_p \perp_p = a$) and \top_p as its absorbing element ($\forall a \in P, a +_p \top_p = \top_p$); \times_p , the multiplicative operator, is associative, commutative and distributes over $+_p$ ($a \times_p (b +_p c) = (a \times_p b) +_p (a \times_p c)$), with \perp_p as its unit element and \top_p as its absorbing element².

The additive operator of this structure has the same properties as the corresponding one in c-semirings, and thus it induces a partial order over P in the usual way: $a \leq_p b$ iff $a +_p b = b$. This allows to prove that $+_p$ is monotone ($\forall a, b, d \in P$ s.t. $a \leq_p b, a +_p d \leq_p b +_p d$).

² In fact, the absorbing nature of \top_p can be derived from the other properties.

$b \times_p d$) and that it is the least upper bound in the lattice (P, \leq_p) ($\forall a, b \in P, a \times_p b \geq_p a +_p b \geq_p a, b$).

On the other hand, \times_p has different properties w.r.t. \times_n : the best element in the ordering (\top_p) is now its absorbing element, while the worst element (\perp_p) is its unit element. \perp_p models indifference. These are exactly the desired properties for the combination and for indifference w.r.t. positive preferences. An example of a positive preference structure is $(\mathbb{R}^+, max, sum, 0, +\infty)$, where preferences are positive real numbers aggregated with *sum* and compared with *max*.

5 BIPOLAR PREFERENCE STRUCTURES

For handing both positive and negative preferences we propose to combine the two structures described in sections 4 and 3 in what we call a *bipolar preference structure*. A bipolar preference structure is a tuple $(N, P, +, \times, \perp, \square, \top)$ where, $(P, +|_P, \times|_P, \square, \top)$ is a positive preference structure; $(N, +|_N, \times|_N, \perp, \square)$ is a c-semiring; $+ : (N \cup P)^2 \rightarrow (N \cup P)$ is an operator s. t. $a_n + a_p = a_p$, $\forall a_n \in N$ and $a_p \in P$; it induces a partial ordering on $N \cup P$; $\forall a, b \in P \cup N, a \leq b$ iff $a + b = b$; $\times : (N \cup P)^2 \rightarrow (N \cup P)$ is a commutative and monotone ($\forall a, b, c \in N \cup P$, if $a \leq b$, then $a \times c \leq b \times c$) operator.

Bipolar preference structures generalize both c-semirings and positive structures. In fact, when $\square = \top$, we have a c-semiring and, when $\square = \perp$, we have a positive structure. Given the way the ordering is induced by $+$ on $N \cup P$, easily, we have $\perp \leq \square \leq \top$. Thus, there is a unique maximum element (that is, \top), a unique minimum element (that is, \perp); the element \square is smaller than any positive preference and greater than any negative preference, and it is used to model indifference.

A bipolar preference structure allows to have a richer structure for one kind of preference, that is common in real-life problems. In fact, we can have different lattices (P, \leq_p) and (N, \leq_n) . In the following, we will write $+_n$ instead of $+|_N$ and $+_p$ instead of $+|_P$. Similarly for \times_n and \times_p . When \times is applied to a pair in $(N \times P)$, we will sometimes write \times_{np} and we will call it compensation operator.

An example of bipolar structure is the tuple $(N=[-1, 0], P=[0, 1], +=\max, \times, \perp=-1, \square=0, \top=1)$, where \times is such that $\times_p = \max$, $\times_n = \min$ and $\times_{np} = \text{sum}$. Negative preferences are between -1 and 0, positive preferences between 0 and 1, compensation is sum, and the order is given by max. In this case \times is not associative.

In general, operator \times may be not associative. For example, if $\top \times \perp = c \in (N \cup P) - \{\top, \perp\}$ or if there are at least two elements $p \in P - \{\top\}, n \in N - \{\perp\}$ s.t. $p \times n = \square$ and \times_n or \times_p is idempotent, then \times is not associative. Since these conditions often occur in practice, it is not reasonable to require associativity of \times .

The combination of a positive and a negative preference is a preference which is higher than, or equal to, the negative one and lower than, or equal to, the positive one.

Possible choices for combining strictly positive with strictly negative preferences are thus the average, the median, the min or the max operator. Moreover, by monotonicity, if $\top \times \perp = \perp$, then $\forall p \in P, p \times \perp = \perp$. Similarly, if $\top \times \perp = \top$, then $\forall n \in N, n \times \top = \top$.

6 BIPOLAR PREFERENCE PROBLEMS

Once we have defined bipolar preference structures, we can define a notion of bipolar constraint, which is just a constraint where each assignment of values to its variables is associated to one of the elements in a bipolar preference structure. Given a bipolar preference structure

$(N, P, +, \times, \perp, \square, \top)$ a finite set D (the domain of the variables), and an ordered set of variables V , a constraint is a pair $\langle def, con \rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow (N \cup P)$.

A bipolar CSP (V, C) is then just a set of variables V and a set of bipolar constraints C over V .

A solution of a bipolar CSP (V, C) is a complete assignment to all variables in V , say s , with an associated preference $pref(s) = (p_1 \times_p \dots \times_p p_k) \times (n_1 \times_n \dots \times_n n_l)$, where, for $i := 1, \dots, k$ $p_i \in P$, for $j := 1, \dots, l$ $n_j \in N$, and $\exists \langle def, con \rangle \in C$ such that $p_i = def(s|_{con})$ or $n_j = def(s|_{con})$. A solution s is optimal if there is no other solution s' with $pref(s') > pref(s)$. In this definition, the preference of a solution s is obtained by combining all the positive preferences associated to its projections over the constraints, combining all the negative preferences associated to its projections over the constraints, and then, combining the two preferences obtained so far. If \times is associative, then other definitions of solution preference could be used while giving the same result.

7 RELATED AND FUTURE WORK

Bipolar reasoning and preferences have recently attracted some interest in the AI community. In [1], a bipolar preference model based on a fuzzy-possibilistic approach is described, but positive and negative preferences are kept separate and no compensation is allowed. In [6] only totally ordered unipolar and bipolar preference scales are used, while we have presented a way to deal with partially ordered ones. On totally ordered scales the t-norm and t-conorm of [6] correspond to \times_n and \times_p , while the uninorm of [6], similar to \times , is less general than \times , since it is associative.

We plan to adapt constraint propagation and branch and bound techniques to deal with bipolar problems and we intend to develop a solver for bipolar CSPs, which should be flexible enough to accommodate for both associative and non-associative compensation operators. We also intend to consider the presence of uncertainty in bipolar problems, possibly using possibility theory and to develop solving techniques for such scenarios. We also want to generalize other preference formalisms, such as multicriteria methods and CP-nets.

8 ACKNOWLEDGEMENTS

This work has been supported by Italian MIUR PRIN project “Constraints and Preferences” (n. 2005015491).

REFERENCES

- [1] S. Benferhat, D. Dubois, S. Kaci, and H. Prade, ‘Bipolar representation and fusion of preferences in the possibilistic logic framework’, in *KR-02*. Morgan Kaufmann, (2002).
- [2] S. Bistarelli, U. Montanari, and F. Rossi, ‘Semiring-based constraint solving and optimization’, *Journal of the ACM*, **44**(2), 201–236, (1997).
- [3] S. Bistarelli, M. S. Pini, F. Rossi, and K. B. Venable, ‘Positive and negative preferences’, in *CP 2005 workshop on preferences and soft constraints*, (2005).
- [4] R. Dechter, *Constraint processing*, Morgan Kaufmann, 2003.
- [5] H. Fargier, T. Schiex, and G. Verfaillie, ‘Valued Constraint Satisfaction Problems: Hard and Easy Problems’, in *IJCAI-95*, pp. 631–637. Morgan Kaufmann, (1995).
- [6] M. Grabisch, B. de Baets, and J. Fodor, ‘The quest for rings on bipolar scales’, *Int. Journ. of Uncertainty, Fuzziness and Knowledge-Based Systems*, (2003).

A Connectivity Constraint using Bridges

Patrick Prosser¹ and Chris Unsworth²

1 Introduction

We present a specialised constraint for enforcing graph connectivity. It is assumed that we have a square symmetrical array A of 0/1 constrained integer variables representing potential undirected edges in a simple graph, such that variable $A[u, v]$ corresponds to the undirected edge (u, v) . A search process is then at liberty to select and reject edges. The connectivity constraint ensures that no *critical* edge may be deleted from A , i.e. an edge that if deleted disconnects the graph. This connectivity constraint might then be used when modelling network design problems, modelling molecular structures, problems in bioinformatics [4], or graph problems where connectivity is an essential property [3, 1].

The constraint's internals are founded on the depth first search (dfs) process. The dfs process generates two sets of edges: the set of tree edges T , and the set of back edges B . The connectivity constraint associates a counter $nc[u, v]$, initialised to zero, with each tree edge. On creation of a back edge (t, w) we increment the counters on the tree edges spanned by that back edge, i.e. we increment the set of counters $\{nc[u, v] \mid (u, v) \in path(t, w)\}$ where $path(t, w)$ delivers the set of edges that make up the path from t to w in T . A counter is incremented to indicate that there is at least one more cycle in the graph involving edge (u, v) , i.e. $nc[u, v]$ is a count of the number of cycles passing through edge (u, v) using edges in T and B . If on termination of the dfs process any counter, say $nc[v, w]$, equals zero then that edge (v, w) is a bridge (also known as an isthmus or cut-edge) and must be forced. Consequently the value 0 is removed from the domain of variable $A[u, v]$.

2 Depth First Search with Tree and Back Edges

We assume we have a graph G with vertices V and edges E . The dfs algorithm [2] will produce two sets of edges: T the tree edges, and B the back edges. A vertex may be given one of three colours: white, grey, or black. Initially all vertices are white, signifying that they have not been visited. A vertex v is coloured grey when it has first been visited, i.e. a call to $dfs(v)$ is made. On completing the call to $dfs(v)$ the vertex is coloured black. Therefore a vertex starts white, may then turn grey, and eventually black. In the dfs algorithm, given below, two types of edges may be created.

- (a) Edge (v, w) is a tree edge when v is adjacent to w in G and v is grey and w is white. In the algorithm the call $treeEdge(v, w)$ adds the edge (v, w) to T . A tree edge is created as dfs expands forwards to a new descendant.
- (b) A back edge (v, w) is created when w is adjacent to v , w has already been visited by dfs (i.e. w is grey), the processing of v

is not yet complete (i.e. v is grey), and (w, v) is not already a tree edge. The call $backEdge(v, w)$ does nothing if $(w, v) \in T$, otherwise the edge (v, w) is added to B and the set of counters $\{nc[x, y] \mid (x, y) \in path(v, w)\}$ are incremented, i.e. increment the cycle counters on the tree edges on the path from v to w .

```

procedure dfs(v)
begin
  colour[v] := grey;
  for w in adjacent(v)
  do begin
    if colour[w] = white
    then begin
      treeEdge(v, w)
      dfs(w)
    end
    else if colour[w] = grey
    then backEdge(v, w)
  end
  colour[v] := black;
end

```

3 The Methods of the Constraint

We now present the methods that act upon the constraint. First we describe the actions that take place when the constraint is initially added to a model. Then we describe the methods that are performed when an edge is rejected by the search process. Note that no action need be taken when an edge is selected by the search process as this cannot result in the graph becoming disconnected.

3.1 On Awakening

When the constraint is initially awoken (i.e. added to the model) a dfs is performed from an arbitrary vertex. As noted above we associate cycle counters with each tree edge, i.e $nc[u, v]$ is a count of the number of cycles through the edge (u, v) resulting from back edges and tree edges that span or include edge (u, v) . These counters are initially zero and are incremented when spanned by a back edge. On termination of dfs, any cycle counter $nc[u, v]$ equal to zero indicates that edge (u, v) is a bridge in G and that edge must be selected otherwise G will be disconnected. In addition, if on completion of the call to dfs any vertex is white then that vertex is isolated and the graph cannot be connected; consequently we can raise an exception.

3.2 On Deletion of a Back Edge

If an edge (u, v) is deleted from the graph and (u, v) is a back edge then we decrement the set of counters $\{nc[x, y] \mid (x, y) \in path(u, v)\}$, i.e. decrement all cycle counters on the path from u to v in T . If any counter reaches zero then the corresponding edge is a bridge and must be forced.

¹ Department of Computing Science, University of Glasgow, Scotland, pat@dcs.gla.ac.uk

² Department of Computing Science, University of Glasgow, Scotland, chrisu@dcs.gla.ac.uk

3.3 On Deletion of a Tree Edge

If a tree edge (u, v) is deleted a new subtree must be produced and one of the back edges spanning (u, v) will become a tree edge. Such a candidate back edge must exist otherwise (u, v) will have been a bridge and that bridge will have been detected and its selection already forced. The following actions (also shown pictorially in Figure 1) need to be performed when tree edge (u, v) is deleted where we assume that u is the parent of v :

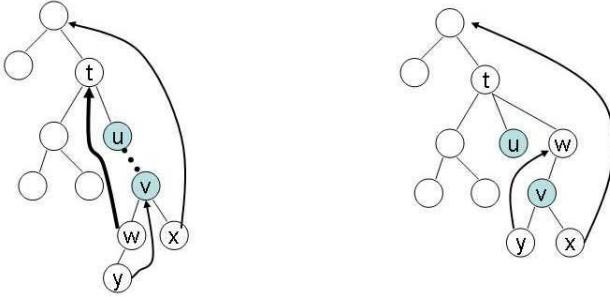


Figure 1. Tree edge (u, v) is deleted on the left and repaired on the right.

1. Let V' be the set of vertices in the subtree rooted on v (and includes v), T' the set of tree edges in that subtree, and B' the set of back edges $\{(t, w) \mid (t, w) \in B \wedge w \in V'\}$.
2. Decrement the *multi-set* of cycle counters $\{\{nc[x, y] \mid (x, y) \in path(t, w) \wedge (t, w) \in B'\}\}$, where $path(t, w)$ delivers the set of edges on the path from t to w in T .
3. Find the back edge $(t, w) \in B'$ where $depth(t) \leq depth(u) < depth(v) < depth(w)$ and $depth(t)$ is a maximum. (The back edge (t, w) is shown on the left of Figure 1).
4. Colour the vertices in the set V' white (i.e. mark them as not visited by dfs).
5. Remove the tree edges T' and back edges B' , i.e. $T = T \setminus T'$ and $B = B \setminus B'$.
6. Colour grey the set of vertices on the path from t to the root of T . Note, that if this was not done then no new back edges could be produced involving ancestors of t in step 7 below.
7. Add new tree edge (t, w) to T , i.e. what was back edge (t, w) becomes a tree edge. Now make a call to $dfs(w)$. (The repaired subtree is shown on the right of Figure 1).
8. Colour black the set of vertices on the path from t to the root of T . This needs to be done to prevent forward edges being produced by subsequent calls to dfs
9. If any cycle counter $nc[x, y]$, where $(x, y) \in T$, is zero then the edge is a bridge and must be selected.

Note that in step 3 we must select the deepest spanning back edge otherwise a cross edge may be produced in step 7 and the cycle counters would be corrupted, and that such a back edge must exist. On the termination of step 7 there will be no white vertices. This could only happen if edge (u, v) was a bridge, and that would be a contradiction.

4 Complexity

The complexity of the algorithm is $O(n^3)$ for a graph with n vertices. The principal activity of the algorithm is the number of times the cycle counters are incremented. The worst case graph would be the clique K_n . A clique would have a dfs tree with back edges as shown in Figure 2, using K_6 as an example. As can be seen, the vertices of the graph have been linearised, and the remaining edges in the clique

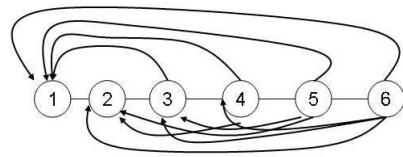


Figure 2. A dfs tree (straight lines) for K_6 with back edges (curved lines). The sum of the number of times edges participate in cycles is $\frac{n^3 - 7n}{6} + 1$

become back edges. The tree has $n - 1$ tree edges, and therefore $n - 1$ cycle counters. A tree edge emanating from a vertex at position i (where the first position is $i = 1$ and the last $i = n - 1$) will be involved in at most $i(n - i) - 1$ cycles. That is, for a vertex at depth i there will be $n - i$ vertices below it. Each one of these $n - i$ vertices can then have back edges to each of the first i vertices. However, the vertex in position $i + 1$ cannot have a back edge to the vertex in position i , otherwise we have a cycle that involves only 2 vertices. Therefore we must remove 1 from our calculation. Consequently the sum of the cycle counters on the dfs tree T for the clique K_n will be as follows:

$$\sum_{i=1}^{n-1} [i(n - i) - 1] = \frac{n^3 - 7n}{6} + 1$$

We should expect that the performance of this can be improved by taking a lazy approach, possibly based on [5].

5 Data Structures and Potential Enhancements

A number of reversible data structures are required to realise the above. Since we need to traverse a subtree in order to delete tree and back edges we associate with each vertex a boolean set of length n , representing the immediate children of that vertex. We also associate with a vertex a boolean set of length n representing the back edges from a vertex. Each vertex also has a parent attribute so that we can traverse from a vertex upwards towards some other vertex downgrading the cycle counters. We also associate a depth with a vertex so that we can compare back edges. All of the above additional space is of order $O(n^2)$ where n is the number of vertices in G .

The constraint can be enhanced to deal with multi-edges. Assuming that an array variable $A[u, v]$ can have a value greater than 1, if the constraint forces the edge (u, v) rather than setting $A[u, v]$ to 1 we remove the value 0. The efficiency of step 2 in 3.3 can be improved by setting counters in the tree edges rooted on v to zero, and decrementing counters on tree edges above u .

REFERENCES

- [1] Ken Brown, Patrick Prosser, J. Christopher Beck, and Christine Wu, ‘Exploring the use of constraint programming for enforcing connectivity during graph generation’, in *The 5th workshop on modelling and solving problems with constraints (held at IJCAI05)*, (2005).
- [2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, Sept 2001.
- [3] Gregoire Dooms, Yves Deville, and Pierre Dupont, ‘CP(Graph): Introducing a Graph Computation Domain in Constraint Programming’, in *CP2005 (LNCS 3709)*, (2005).
- [4] Ian P. Gent, Patrick Prosser, Barbara M. Smith, and Christine Wu Wei, ‘Supertree Construction with Constraint Programming’, in *CP2003 (LNCS 2833)*, (2003).
- [5] R. Endre Tarjan, ‘A note on finding the bridges of a graph’, *Information Processing Letters*, 2, 160–161, (1974).

Acknowledgements: We would like to thank our five reviewers.

3. Distributed AI/Agents

This page intentionally left blank

Search Better and Gain More: Investigating New Graph Structures for Multi-Agent Negotiations

Samir Aknine¹

Abstract. Combined negotiation in multi-agent systems is a hard task since it involves several levels of difficulty. In order to improve their payoff, first agents behave in a strategic manner while bargaining since they need to deal with various types of behaviors. Second agents have to react to the proposals of other agents in finding the optimal solutions for their negotiation. This paper tackles the problem of winner determination in combined multi-agent negotiations and addresses two fundamental issues. The first contribution of this work is a winner determination algorithm, which finds an optimal solution, which is a combination of several bids selected from a set of bids at one iteration of a combined negotiation. In addition, the problem of dynamically revising these optimal solutions with regards to changes on bids is considered. These changes occur in multi-agent negotiation processes having several iterations and which use multi-phased protocols. To date no work addresses this problem. The second contribution of this work is an iterative algorithm for updating the optimal solutions to avoid integral and repetitive reapplication of the winner determination algorithm. The results of the experiments carried out using our algorithms have confirmed the importance and the originality of our approach based on the use of shared and unshared item graphs.

1 INTRODUCTION

This paper addresses the optimal winner determination problem in a multi-agent combined negotiation and the dynamic revision of the optimal solution. It proposes a search and update algorithm for optimal solutions. Currently, several algorithms [2,3,5,6] determine the best bid or combination of bids maximizing the utility function of an agent during an iteration of a combined negotiation. However, the problem of combined negotiation is that the search for an optimal solution is repetitive since in each iteration or negotiation phase negotiator agents must take a decision and select their new optimal solution if the bids formulated in preceding iterations have changed. Our aim is to propose an algorithm to find this optimal solution incrementally. To date no work addresses this problem.

An optimal solution is recomputed from one iteration to another, and depending on the utility reached by the agents with this solution the negotiation is stopped or resumed. The incremental searching requires an algorithm able to update the optimal solution with minimum computations in order to reduce its complexity. Such an algorithm will compute a new solution by taking into account only the main changes that have appeared in prior iterations.

¹ LIP6, Université Paris 6, Samir.Aknine@lip6.fr

The types of modification to be considered depend on the evolution of the bids. For instance, if a bid at stake becomes invalid – its sender has retracted from the negotiation – this requires an operation to delete this bid and all the solutions using these bids must be revisited. If a bidder sends a new bid, it will imply an operation to complete the list of bids and possibly the solutions. Other changes on the bids may also happen if the bidders decide to increase or decrease their bids. Remember that all these behaviors are feasible in a multi-agent negotiation.

Regarding these new constraints and knowing that the optimal winner determination problem is *NP-complete*, it is not possible to reapply the basic algorithm for optimal solution search at each iteration of a negotiation. An optimal solution should be recomputed in each step of the negotiation but with an algorithm requiring a minimum of computations in order to reduce computational complexity, since in a negotiation based on several phases this search for new solutions could be reiterated as many times as the negotiation requires. Thus the algorithm has to find this new optimal solution by considering the changes that have occurred on the bids.

To address this problem, our approach recommends building two graphs, named shared and unshared item graphs, before building the optimal solution tree for the winner determination problem. These graphs enable us to work with an intentional representation of bids rather than an extensional representation as is done in current algorithms. They provide the sufficient and necessary conditions for the analysis of a bid or a partition of bids. We prove that this method reduces the exploration runtime of the bids. To build the optimal solution tree, these graphs and their properties can be combined with any method for tree construction (depth-first, width-first or mixed, etc.). We have chosen to base our method on certain principals of [5,6,7] for the construction of trees also based on the IDA* method of [4]. We show that our method finds optimal solutions for combined negotiations in a shorter time and explores less search space. These improvements are due to the graph structures we propose.

2 WINNER DETERMINATION

Formally, the winner determination problem in combined negotiations is such that: having a buyer agent which proposes to acquire $L = \{1, 2, 3, \dots, n\}$ items, where each item is defined by a set of properties, for instance the price; and a group of seller agents who propose a set $E = \{O_1, O_2, O_3, \dots, O_m\}$ of bids, where each O_i is a subset of items in L called its structure. A bid is defined by a set of properties such as the price v_i , for instance.

The number of items included in the bid indicates its length. Each item can appear in several bids. We deal with only one property for items, their price, in this algorithm. [5,6,7] have presented several algorithms to solve the winner determination problem. In this paper we essentially focus on the algorithm for multi-agent combined negotiation. Our approach recommends looking first for the similarities and differences between all the various bids in order to reuse them later as well as possible in the search phase for the optimal solution, since we already know that the problem of optimal solution search is *NP-complete* [1,2]. We deal with these properties in the graphs below.

Definition 1 *A shared item graph is a connected directed acyclic graph organized in several levels. First level nodes in this graph are the partitions resulting from the classification of the bids. Intermediate nodes contain items shared by the nodes they connect, i.e. the connected nodes contain either bids, as in the nodes of the first level, or items as in the nodes of intermediate levels. The terminal node of the graph is empty.*

Parallel to the construction of the shared item graph, our approach recommends the construction of the unshared item graph. This second graph highlights the differences between the bids of the partitions, and is defined as follows.

Definition 2 *An unshared item graph is a connected directed acyclic graph organized in several levels. The first level nodes in this graph are the partitions of bids. Each node of the second level contains the items not shared by all the bids of the partition to which this node is connected. The next intermediate nodes contain items shared by the nodes they connect to. The terminal node of the unshared item graph is empty.*

3 REVISING THE OPTIMAL SOLUTIONS

As mentioned previously, several changes may occur from one iteration of the negotiation to another: (1) withdrawals of bids; (2) changes of the properties of bids, for instance the price; (3) insertion of new bids. All these changes impact the partitions of bids and the shared and unshared item graphs built in preceding iterations. Depending on the changes carried out on the shared and unshared item graphs, the new search for the optimal solution requires the most restricted set of operations. Remember that at the end of the first search for the optimal solution, the algorithm has built a solution tree. In this new search, the algorithm is partially based on this tree whose trace has, of course, been updated, as have the shared and unshared item graphs.

The tree is then explored in order to add the changes on the bids which were removed or updated and those which have been recently introduced. Note that during the update phase of the partitions, two lists are processed at the same time, i.e. one for new bids L_{New} and one for removed bids L_{Delete} .

When the exploration of a node of the tree indicates a bid which has only been updated, the value of this bid is put in the solution tree. Moreover this value is also changed in the shared and unshared item graphs. To find the position of the bid in these graphs the algorithm uses several properties. If a bid has been removed, before withdrawing the branch under construction the algorithm tries first to rebuild this removed bid

by combining other bids which would possibly belong to L_{New} . If this case cannot be solved, the exploration of the path in the solution tree is stopped. When a bid is removed the partition which contains it could be removed if there are no more bids in this partition. By removing a partition, its ancestor nodes in the shared and unshared item graphs should be revised.

The new bids integrated in the partitions are processed in two different ways. First when a bid is added in a partition, the ancestor nodes in the shared and unshared item graphs are revised except for singleton bids. Then bids added in the first partition involve the construction of new optimal solution subtrees, which are built in exactly the same way as in the first iteration of the optimal solution search [1]. The bids integrated in other partitions are also processed like those in the first partition. For each of these bids a sub-tree is also generated in order to reduce the cost of integrating the new bids in the branches of existing solutions. This approach has proved to be efficient.

4 CONCLUSION

This paper addressed an important issue in multi-agent combined negotiations. Our approach is a continuation of the work on the winner determination problem [2,3,5,6,7]. It is primarily based on three new principles. (1) We use graph structures for shared and unshared items. These graphs enable us to work on an intentional representation of the bids instead of their extensional representation, and this reduces the time needed for the exploration of bids. (2) We defined new functions for ordering bids and items and several properties for the graphs. (3) We also considered the singularity of the items to speed up the search. The proposed algorithm has been implemented, tested on different distributions of bids to evaluate its performance and proved successful.

Acknowledgment. We would like to thank the Master students for all the implementations they performed.

REFERENCES

1. Aknine, S. "Iterated algorithm for the optimal winner determination in combined negotiations", ECAI, 949-450, IOS Press, Valencia, Spain, 2004.
2. Aknine, S. "Improving optimal winner determination algorithms using graph structures", AMEC VI, AAMAS, NewYork, pp.291-304, Springer-Verlag, 2004.
3. Fujishjima Y., Leyton-Brown K. and Shoham Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*, 1999.
4. Korf, R.E. Depth-first iterative-deepening: An optimal admissible tree search, *Artificial Intelligence*, 27 (1), 1985.
5. Sandholm T. Algorithm for optimal winner determination in combinatorial auctions. *AI Journal*, 135:1–54,2002.
6. Sandholm T. and Suri S. BOB: Improved winner determination in combinatorial auctions and generalizations. *AI Journal*, volume 145, pp. 33-58. 2003.
7. Sandholm T., Suri S., Gilpin A. and Levine D. CABOB: A fast optimal algorithm for combinatorial auctions. *IJCAI*, Seattle, August, 2001.

An argumentation-based framework for designing dialogue strategies

Leila Amgoud¹ and Nabil Hameurlain²

Abstract. A dialogue strategy is the set of rules followed by an agent when choosing a move (act + content) during a dialogue. This paper argues that a strategy is decision problem in which an agent selects i) among the acts allowed by the protocol the best option that, according to some *strategic beliefs* of the agent will at least satisfy the most important *strategic goals* of the agent, and ii) among different alternatives (eg. different offers), the best one that according to some *basic beliefs* of the agent, will satisfy the *functional goals* of the agent. The paper proposes a formal framework based on argumentation for computing the best move to play at a given step of the dialogue.

1 INTRODUCTION

A dialogue protocol identifies the different possible replies after a given act. However, the exact act to utter at a given step of the dialogue is a *strategy* matter. A strategy can be seen as a two steps *decision process*: i) among all the possible replies allowed by the protocol, to choose the move to play. ii) to choose the content of the move if any. In most works on modeling dialogues, the definition of a protocol poses no problems. However, the situation is different for dialogue strategies. There is no formal models for defining them. There is even no consensus on the different ingredients involved when defining a strategy. This paper argues that the strategy is a *decision problem* in which an agent tries to choose among different *alternatives* the best option, which according to its beliefs, will satisfy at least its most important goals. Two kinds of goals (resp. of beliefs) are distinguished: the *strategic* and the *functional* goals (resp. the *strategic* and *basic* beliefs). The strategic goals are the meta level goals of the agent. Such goals will help an agent, on the basis of the strategic beliefs, to select the type of act to utter. Regarding functional goals, they will help an agent to select, on the basis of the basic beliefs, the content of a move.

We propose a formal argumentation-based framework for defining strategies. This framework takes as input two sets of goals: the strategic and the functional goals together with the strategic and basic beliefs and returns among the possible replies allowed by the protocol after a given act, the next move (act + its content) to play. The model is an extension of the argument-based decision framework proposed in [1]. The basic idea behind this model is to construct for each alternative the different arguments (reasons) supporting it, then to use a *criterion* to compare pairs of alternatives on the basis of the quality of their supporting arguments.

¹ IRIT - CNRS, 118 route de Narbonne 31062 Toulouse Cedex 09, France
email: amgoud@irit.fr

² LIUPPA-Université de Pau, Avenue de l'Université 64012 Pau Cedex email:
nabil.hameurlain@univ-pau.fr

2 AGENT'S MENTAL STATES

Two different kinds of goals are involved when selecting the next act to play and its content:

Strategic goals: For choosing the type of act to utter, an agent refers to what we call *strategic goals*. By strategic goals we mean the meta-level goals of the agent such as “minimizing the dialogue time”, “selling at the end of the dialogue”, etc. Suppose that at a given step of a negotiation dialogue, an agent has to choose between making an offer and asking a question. If the agent wants to minimize the dialogue time then it would choose to make an offer instead of spending more time in questions. However, if the agent wants to get a maximum of information about the wishes of the other agent, then the agent would decide to ask a question. These goals are generally independent of the *subject* of the dialogue.

Functional goals: The goals of the agent which are directly related to the subject of the dialogue are called *functional goals*. They represent what an agent wants to achieve or to get regarding the subject of the dialogue. Let us take the example of the agent negotiation the place of a meeting. The agent may prefer a place which is not warm and not expensive. These functional goals are involved when selecting the content of a move. In a negotiation, an agent proposes offers that satisfy such goals.

As for goals, the beliefs involved in the two decision problems (selecting an act and a content) are also of different nature:

Strategic beliefs: which are the meta-level beliefs of the agent. They may represent the beliefs of the agent about the dialogue and about the other agents involved in the dialogue. In negotiation dialogues where agents are trying to find a common agreement, agents may intend to simulate the reasoning of the other agents. Thus it is important for each agent to consider the beliefs that it has on the other agents' goals and beliefs.

Basic beliefs: represent the beliefs of the agent about the environment and the subject of the dialogue. Let us consider again the example of the agent negotiating the place of a meeting. Basic beliefs of the agent may include for instance the fact that “London is not warm”, “Tunisia is hot”, “London is very expensive”, etc.

3 THE LOGICAL LANGUAGE

Let \mathcal{L} be a propositional language and $Wff(\mathcal{L})$ the set of well-formed formulas built from \mathcal{L} . Let us suppose an additional symbol \bot . Each agent has the following bases:

$\mathcal{B}_f = \{(k_p, \rho_p), p = 1, s\}$ where $k_p \in Wff(\mathcal{L})$, is the basic beliefs. The beliefs are associated with certainty levels ρ_p . A pair (k_p, ρ_p) means that k_p is at least certain at a degree ρ_p .

$\mathcal{B}_s = \{(l_j, \delta_j), j = 1, m\}$ where $l_j \in Wff(\mathcal{L})$ is the strategic beliefs base. Each of these beliefs has a certainty level δ_j .

$\mathcal{G}_s = \{(g_q, \lambda_q), q = 1, t\}$ where $g_q \in Wff(\mathcal{L})$, is a base of strategic goals. The strategic goals can have different priority degrees, represented by λ_q . A pair (g_q, λ_q) means that the goal g_q is important for the agent at least to a degree λ_q .

$\mathcal{G}_f = \{(go_r, \gamma_r), r = 1, v\}$ where $go_r \in Wff(\mathcal{L})$, is the base of the functional goals of the agent. Each functional goal has a degree of importance denoted by γ_r .

The different certainty levels and priority degrees are assumed to belong to a unique linearly ordered scale with maximal element denoted by 1 (corresponding to total certainty and full priority), and a minimal element denoted by 0 corresponding to the complete absence of certainty or priority. We shall denote by \mathcal{B}_f^* , \mathcal{B}_s^* , \mathcal{G}_s^* , \mathcal{G}_f^* the corresponding sets when weights are ignored.

Let \mathcal{S} be the set of speech acts allowed by the protocol. Let us suppose that the function $\text{Replies}: \mathcal{S} \rightarrow 2^{\mathcal{S}}$ returns for each act, all the legal replies to it. Some acts may have a content. For instance, an act **Offer** should be accompanied with a content such as a price, a town, etc. However, the act **Withdraw** does not need any content. Such acts will have an empty content, denoted by the symbol ?. The function $\text{Content}: \mathcal{S} \rightarrow \mathcal{L} \cup ?$ returns for a given act the set of its possible contents. During a dialogue, agents exchange *moves* which are pairs (a, x) , where $a \in \mathcal{S}$ and $x \in \text{Content}(a)$. The strategy problem is formalized as follows: let (a, x) be the current move in a dialogue. What is the next move (a', x') to utter such that $a' \in \text{Replies}(a) ?$ To answer this question, one should find both a' and x' . Indeed, a' is the “best” element in $\text{Replies}(a)$ that satisfies \mathcal{G}_s^* according to \mathcal{B}_s^* , whereas x' is the “best” element among $\text{Content}(a')$ that satisfies \mathcal{G}_f^* according to \mathcal{B}_f^* .

4 THE ARGUMENTATION-BASED DECISION MODEL

Solving a decision problem amounts to defining a pre-ordering, usually a complete one, on a set \mathcal{X} of possible choices, on the basis of the different consequences of each decision. In our case, the set \mathcal{X} may be either the set $\text{Replies}(a)$ of the possible replies to a move, or the set $\text{Content}(a')$ with a' is the chosen act. Recently, Amgoud [1] has shown that argumentation can provide such a pre-ordering. Let \mathcal{B} be any *consistent* and *weighted* beliefs base, and \mathcal{G} be a *consistent* prioritized base of goals. \mathcal{B}^* and \mathcal{G}^* will denote the corresponding sets when weights are ignored. From these bases, different arguments in favor of elements of \mathcal{X} can be built. The idea is that a decision is justified and supported if it leads to the satisfaction of at least the most important goals of the agent, taking into account the most certain part of knowledge. An *argument in favor of* a choice $x \in \mathcal{X}$ is a triple $A = \langle S, g, x \rangle$ such that $S \subseteq \mathcal{B}^*$, $g \in \mathcal{G}^*$, $S \cup \{x\}$ is consistent, $S \cup \{x\} \vdash g$, and S is minimal (for set inclusion) among the sets satisfying the above conditions. S is the *support* of the argument, g is the *goal* which is reached by the decision x and x is the *conclusion* of the argument. The set $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$ gathers all the arguments which can be constructed from \mathcal{B} , \mathcal{G} and \mathcal{X} . $\text{Arg}(x)$ is the set of arguments whose conclusion is x .

Property 1 If $\mathcal{B} = \emptyset$, or $\mathcal{G} = \emptyset$, then $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X}) = \emptyset$.

Since an argument involves two kinds of information (beliefs and goals), its strength depends on the quality of both information. Thus, to each argument we associate a certainty level representing the degree of the less entrenched belief used in it, and a weight representing the importance of the goal reached by the conclusion of that

argument. The *strength* of an argument $A = \langle S, g, x \rangle$ is a pair $\langle \text{Level}(A), \text{Weight}(A) \rangle$ such that: i) $\text{Level}(A) = \min\{\rho_i \mid k_i \in S \text{ and } (k_i, \rho_i) \in \mathcal{B}\}$. If $S = \emptyset$ then $\text{Level}(A) = 1$. It represents the *certainty level* of the argument. ii) $\text{Weight}(A) = \lambda$ with $(g, \lambda) \in \mathcal{G}$. It represents the *degree of satisfaction* of the argument. The strengths of arguments make it possible to compare pairs of arguments in several ways, depending on whether the agent gives the same importance to the certainty level and the weight or not. For the purpose of illustration, let us consider the following particular criterion in which goals take precedence over beliefs: let A and B be two arguments in $\mathcal{A}(\mathcal{B}, \mathcal{G})$. A is *preferred* to B , denoted $A \succeq B$, iff $\text{Weight}(A) > \text{Weight}(B)$, or $\text{Weight}(A) = \text{Weight}(B)$ and $\text{Level}(A) \geq \text{Level}(B)$. At the core of our framework is the use of a *principle* that allows for an argument-based comparison of alternatives. Each combination of a principle with a criterion for comparing arguments captures a particular agent *profile*, thus a particular strategy for selecting moves in a dialogue.

Below we present an example of a principle that takes as input the supporting arguments and prefers an alternative which has at least one supporting argument which is preferred to any supporting argument of the other alternative. Formally, let $x, x' \in \mathcal{X}$. x is preferred to x' , denoted $x \triangleright x'$, iff $\exists P \in \text{Arg}(x) \text{ s.t. } \forall P' \in \text{Arg}(x'), P \succeq P'$.

Property 2 \succeq and \triangleright are total preorder relations i.e. reflexive and transitive.

An *argumentation-based decision framework* is a tuple $\langle \mathcal{X}, \mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X}), \succeq, \triangleright \rangle$ where \mathcal{X} is a set of all possible decisions, $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$ is a set of arguments built from \mathcal{B} , \mathcal{G} and \mathcal{X} , \succeq is a (partial or complete) pre-ordering on $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$, and \triangleright defines a (partial or complete) pre-order on \mathcal{X} . This framework can be used for move selection. Indeed, let \mathcal{X} be a set of possible decisions. The set of best decisions is $\text{Best}(\mathcal{X}) = \{x \in X, \text{s.t. } \forall x' \in \mathcal{X}, x \triangleright x'\}$.

Property 3 If $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X}) = \emptyset$, then $\text{Best}(\mathcal{X}) = \emptyset$.

Now, let (a, x) be the current move of the dialogue, and an agent has to choose the next one, say (a', x') . The act a' is returned as a best option by the framework $\langle \text{Replies}(a), \mathcal{A}(\mathcal{B}_s, \mathcal{G}_s, \text{Replies}(a)), \succeq, \triangleright \rangle$ (i.e. $a' \in \text{Best}(\text{Replies}(a))$), whereas the content x' is among the best options returned by the framework $\langle \text{Content}(a'), \mathcal{A}(\mathcal{B}_f, \mathcal{G}_f, \text{Content}(a')), \succeq, \triangleright \rangle$, i.e. $x' \in \text{Best}(\text{Content}(a'))$.

Property 4 If $\mathcal{G}_s = \emptyset$, or $\mathcal{B}_s = \emptyset$, then the next move is $(?, ?)$ meaning that there is no rational solution.

5 CONCLUSION

This paper claims that during a dialogue, a strategy is used only for defining the next move to play at each step of the dialogue. This amounts to define the speech act to utter and its content if necessary. The strategy is then regarded as a two steps *decision process*: among all the replies allowed by the protocol, an agent should select the best speech act to play, then it should select the best content for that speech act. We have shown that the choice of the next speech act is based on the strategic beliefs and the strategic goals, whereas the choice of the content is based on the basic beliefs and the functional goals. We have then proposed a formal framework for defining strategies based on argumentation. We have shown also the agents profiles play a key role in defining principles for comparing decisions.

REFERENCES

- [1] L. Amgoud, ‘A general argumentation framework for inference and decision making’, in *Proceedings of UAI*, pp. 26–33, (2005).

A Multiagent System for Scheduling Activities Onboard a Space System

Francesco Amigoni and Simone Fare¹ and Michèle Lavagna and Guido Sangiovanni²

Abstract. The possible advantages of employing multiple agents to manage activities on a single space system are largely unexplored. This paper presents the experimental validation of a multiagent scheduler for a low Earth orbit satellite.

1 INTRODUCTION

We view a spacecraft as a collection of subsystems, functionally detached but highly interconnected and interdependent in order to reach the goals of the mission. Each subsystem is associated to an agent that is composed of four modules organized in a layered structure [5]: a *planner/negotiator*, a *scheduler*, an *executor*, and a *monitor*. In this paper, we focus on the distributed scheduler.

The separation of functions, the decentralization of activities and responsibilities, and the parallelism enable each agent to quickly answer to local unforeseen events or faults, without involving the overall system. For all these reasons, the multiagent approach is increasingly employed in space applications. However, it is usually applied to formations or constellations of satellites working together [1] or to multi-user scheduling [3]. In our work, we give an original contribution to investigating the use of multiple agents to manage the activities on a *single* space system. This topic is, to the best of our knowledge, largely unexplored (with few exceptions, including the reinforcement learning-based proposal of [4]).

We have developed [6] a multiagent scheduler for *Palamede*, a low Earth orbit satellite under development at the Aerospace Engineering Department of the Politecnico di Milano. The proposed distributed scheduler finds the starting times of the satellite's activities in order to satisfy some temporal and resource constraints.

In this paper, we experimentally show that the multiagent solution to the Palamede's scheduling problems does not introduce any significant loss of performance.

2 THE MULTIAGENT SCHEDULER FOR PALAMEDE

The main task of Palamede is to take pictures of the Earth surface and transmit them to a ground station. The satellite is supposed to orbit around the Earth at a height of approximately 650 km, so one orbit lasts 97 minutes of which 30 minutes are the eclipse period (sun-synchronous midday-midnight orbit).

Our multiagent system for scheduling includes the *Camera* agent, which manages the activities of taking pictures, the *ADCS* agent,

which controls the Attitude Determination and Control System of the satellite, and the *ORBCOMM* agent, which manages the communication flows between Palamede and the ground station via the ORBCOMM transponder. The onboard electrical energy is provided by five body-mounted solar arrays and a Li-Ion battery assembly. By applying a semi-regulated bus system topology, the nominal mode power is provided by the solar arrays; whenever extra power is available, it is used for recharging batteries, nominally discharged during eclipses. To take into account the electrical power consumption in the scheduling activities, a manager agent is associated to this depletable resource: the *Battery* agent. Since the total expected power produced by the solar arrays will be approximately 20 W (excluding the eclipse period) and the onboard data processing subsystem continuously needs 9 W, the management of the power resource is critical, especially during transmission activities that require more than 25 W.

In general, the activities of the agents onboard of Palamede are subject to temporal and resource constraints. These constraints can be either global or local. *Global* constraints involve activities performed by different agents, while *local* constraints involve activities performed by a single agent. *Temporal constraints* determine the temporal ordering in which the activities should be performed. An example of a temporal constraint is *Overlap* between two activities *a* and *b*, which means that *a* must start after *b* has started and must end before *b* ends. For example, in order to take a picture, before a *Grab* (i.e., the action of taking a picture) can be performed, the camera should be on, so *Grab* overlaps *Camera On*. *Resource constraints* determine when it is possible to perform the activities, according to the available onboard resources. The only resource we currently consider is the electrical energy provided by batteries and solar arrays. The resource constraints trivially impose that the power demand should not exceed the power availability. Resource constraints are only global, because electrical power is needed by the activities of all the agents. We decided to introduce the *Battery* agent to handle the aggregate demand of power and to check if it satisfies the resource constraints.

We formulated the scheduling problem as a Distributed Constraint Satisfaction Problem (DCSP):

- there are n sets of activities A_1, A_2, \dots, A_n ; A_i contains the activities of agent i ;
- each activity is associated to a variable representing its starting time, whose time values are taken from a finite domain set H (the scheduling horizon); an activity is characterized also by its duration and its power consumption;
- an agent knows all the temporal constraints (both local and global) relevant to its variables; the resource (global) constraints are known only by the *Battery* agent.

¹ Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy,
email: amigoni@elet.polimi.it, fare@airlab.elet.polimi.it

² Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Italy,
email: lavagna@aero.polimi.it, sangiovanni@aero.polimi.it

In our application, we considered a scheduling horizon H of two orbits (194 minutes), divided in time quanta of 30 seconds. This means that the domain set of a variable (from which the starting time of an activity is selected) is the set of integers from 0 to 387. A solution is an assignment of values to all the variables such that all the temporal and resource constraints are satisfied.

In our multiagent scheduler, the scheduling of the activities is performed at two levels: at the *local level*, in which each agent determines the sequence of its local activities satisfying its known constraints, and at the *global level*, in which the local plans are harmonized to ensure the consistent consumption of shared resources. At the local level, each agent uses the *dynamic backtracking* algorithm [2]. With this algorithm, an agent finds value assignments to its local variables; these assignments form a *local plan*. A local plan of agent i is a vector of values assigned to variables of i , representing the starting times of the activities in A_i . The local algorithm iteratively picks up at random a variable and assigns to it a value chosen according to the least-constraining heuristic. If a value cannot be found, it backjumps to the variable responsible of the failure. Ties are randomly broken. The local plans found at the local level are harmonized at the global level using a variation of the *asynchronous weak commitment* algorithm [7]. The global plan resulting from this cooperation process is a set of local plans that satisfy all the constraints. More details on Palamede's multiagent scheduler are reported in [6].

3 EXPERIMENTAL RESULTS

We have implemented the multiagent scheduler described above in JAVA with the JADE platform. Experiments have been performed with a computer equipped with a 500 MHz processor and 128 MB RAM. In the following, we show that using distributed scheduling on a single spacecraft does not lead to dramatic loss of solution quality (other experiments are reported in [6]). The measures of performance we considered are the fraction of the successful schedules and the time required to reach a consistent global plan.

We initially considered a single agent to schedule h dummy activities with a schedule horizon of two orbits. All the dummy activities do not require any energy to be performed and have the same duration of $T = 5$ time quanta (recall that a time quantum is 30 seconds). The dummy activities are required not to overlap each other by means of the *Overlap* constraint. Of course, given T and the schedule horizon, the larger h , the more difficult the scheduling problem, and the more difficult to find a consistent local plan within the timeout. Fig. 1 shows the performances of the single agent scheduler facing the problem of allocating an increasing number of not overlapping dummy activities. Each point is the average over 100 trials. Variations between different trials are due to random choices in selecting the variable to instantiate, the value for a variable, and the activity to backjump to in the local algorithm. A timeout of 3 minutes has been given to the system for solving each trial: whenever this time limit was reached the trial was stopped and declared unsuccessful. Note that 3 minutes of scheduling is a very short time when compared with the 194 minutes of the scheduling window. Until the number of activities is smaller than 60, the agent instantaneously finds a solution in all the 100 trials. From 60 to 65 activities the agent finds a solution but with a longer average solving time. The computational limit for our local algorithm is reached for $h = 66$, where the solution is found only in the 12% of the trials, within 90 seconds on average. From these results, 60 appears to be the maximum number of activities that a centralized scheduler can easily manage. Hence, we tried different distributions of 60 dummy activities on four agents.

Table 1 shows the performances of the distributed scheduler. In fair situations, with activities symmetrically assigned to agents, the performances are almost the same of the single scheduling agent case. In situations with activities asymmetrically assigned to agents, the loss of successful schedules is slightly more than 10%.

We can conclude that the use of a multiagent system to distribute the scheduling process has not any dramatic impact on the performances of the system but provides the advantage of modularity.

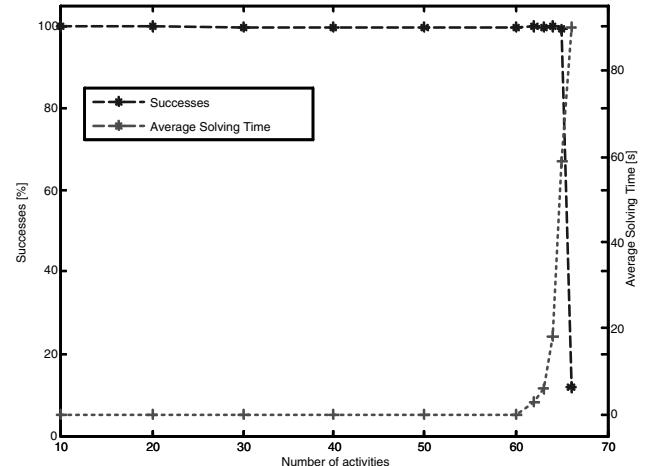


Figure 1. Fraction of successes and average solving time vs. number of activities

Table 1. Distributing activities on four agents.

AGENTS' ACTIVITIES	SUCCESSES [%]				TIME [s]
	1	2	3	4	
15	15	15	15	15	98
20	15	15	10	10	96
30	10	10	10	10	94
35	10	10	5	5	89
40	10	5	5	5	87
45	5	5	5	5	86

REFERENCES

- [1] S. Das, P. Gonsalves, R. Krikorian, and W. Truszkowski, 'Multi-agent planning and scheduling environment for enhanced spacecraft autonomy', in *Proc. Int'l Symposium on Artificial Intelligence, Robotics and Automation in Space*, (1999).
- [2] M. Ginsberg, 'Dynamic backtracking', *AI Research*, **1**, 25–46, (1993).
- [3] J. Jaap and S. Phillips, 'On using an incremental scheduler for human exploration task scheduling', in *Proc. IEEE Aerospace Conf.*, (2005).
- [4] N. Monekosso and P. Remagnino, 'Autonomous spacecraft resource management: A multi-agent approach', in *Proc. Congress of the Italian Association for Artificial Intelligence, LNAI 1792*, pp. 297–308, (2000).
- [5] N. Muscettola, P. Nayak, B. Pell, and B. Williams, 'Remote agent: To boldly go where no ai system has gone before', *ARTIF INTELL*, **103**, 5–47, (1998).
- [6] G. Sangiovanni, S. Farè, F. Amigoni, and M. Lavagna, 'Multiagent-based scheduling and execution of activities for space systems', in *Proc. Int'l Symposium on Artificial Intelligence, Robotics and Automation in Space*, (2005).
- [7] M. Yokoo and K. Hirayama, 'Algorithms for distributed constraint satisfaction: A review', *AUTON AGENT MULTI-AG*, **3**(2), 185–207, (2000).

Benefits of Combinatorial Auctions with Transformability Relationships

Andrea Giovannucci¹ and Jesús Cerquides² and Juan Antonio Rodríguez-Aguilar³

Abstract. In this paper we explore whether an auctioneer/buyer may benefit from introducing his transformability relationships (some goods can be transformed into others at a transformation cost) into multi-unit combinatorial reverse auctions. Thus, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider transformability relationships. Furthermore, we empirically identify the market conditions under which it is worth for the auctioneer/buyer to exploit transformability relationships.

1 Introduction

Consider a company devoted to sell manufactured goods. It can either buy raw goods from providers, transform them into some other goods via some manufacturing process, and sell them to customers; or it can buy already-transformed products and resell them to customers. Thus, either the company buys raw goods to transform via an in-house process at a certain cost, or it buys already-transformed goods. Figure 1 graphically represents an example of a company's inner manufacturing process, more formally *Transformability Network Structure* (TNS), fully described in [1]. This graphical description largely borrows from the representation of Place/Transition Nets (PTN), a particular type of Petri Net [2]. Each circle (corresponding to a PTN place) represents a good. Horizontal bars connecting goods represent manufacturing operations, likewise *transitions* in a PTN. Manufacturing operations are labeled with a numbered t , and shall be referred to as *transformations* (t -relationships henceforth). An arc connecting a good to a transformation indicates that the good is an *input* to the transformation, whereas an arc connecting a transformation to a good indicates that the good is an *output* from the transformation. The labels on the arcs connecting *input goods* to transitions, and the labels on the arcs connecting *output goods* to transitions indicate the units required of each *input good* to perform a transformation and the units generated per *output good* respectively. Each transformation has an associated cost every time it is carried out.

Say that a buying agent requires to purchase a certain amount of goods g_3 , g_5 , g_6 , g_7 , g_8 , g_9 , and g_{10} . For this purpose, it may opt for running a combinatorial reverse auction with qualified providers. But before that, a buying agent may realise that he faces a decision problem: shall he buy g_1 and transform it via an in-house process, or buy already-transformed goods, or opt for a *mixed-purchase* solution and buy some already-transformed goods and some to transform

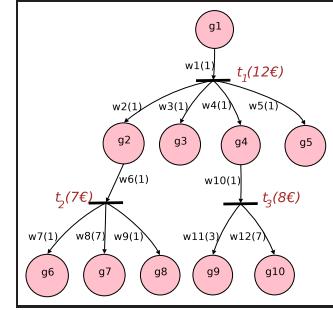


Figure 1. Example of a Transformability Network Structure.
 in-house? This concern is reasonable since the cost of g_1 plus transformation costs may eventually be higher than the cost of already-transformed goods.

The work in [1] addresses the possibility of expressing transformability relationships among the different assets to sell/buy on the bid-taker side in a multi-unit combinatorial reverse auction. The new type of combinatorial reverse auction (the Multi-Unit Combinatorial Reverse Auction with Transformability Relationships among Goods (MUCRAr)) provides to buying agents: (a) a language to express required goods along with the relationships that hold among them; and (b) a winner determination problem (WDP) solver that not only assesses what goods to buy and to whom, but also the transformations to apply to such goods in order to obtain the initially required ones. It is shown that, if the TNS representing the relationships among goods is acyclic, the associated WDP can be modeled by an integer linear program.

The purpose of this paper is twofold. On the one hand, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider transformability relationships. On the other hand, we empirically identify the market conditions under which it is worth for the auctioneer/buyer to exploit transformability relationships. Thus, we provide rules of thumb for an auctioneer/buyer to help him decide when to run a MUCRAr instead of an MUCRA.

2 Empirical Evaluation

Our experiments artificially generate different data sets. Each data set shall be composed of: (1) a TNS; (2) a Request for Quotations (RFQ) detailing the number of required units per good; and (3) a set of combinatorial bids. Then, we solve the WDP for each auction problem regarding and disregarding t -relationships. This is done to quantitatively assess the potential savings that a buyer/auctioneer may obtain thanks to t -relationships, as well as the market conditions where such savings occur. Thus, the WDP for an MUCRA will only con-

¹ IIIA - CSIC, Campus UAB, 08193, Bellaterra, Spain, andrea@iiia.csic.es

² WAI, Dept. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, cerquide@maia.ub.es

³ IIIA - CSIC, Campus UAB, 08193, Bellaterra, Spain, jar@iiia.csic.es

sider the last two components of the data set, whereas the WDP for a MUCRA will consider them all. In order to solve the WDP for an MUCRA we exploit its equivalence with the multi-dimensional knapsack problem [3].

2.1 Experimental Settings and Results

Our goal is to determine under which market conditions MUCRAr leads to savings when compared to MUCRA. At this aim, we empirically measure the differences in outcome cost between MUCRA and MUCRAr. Thus, we define the *Savings Index* (*SI*) as: $SI = 100 \cdot (1 - \frac{C^{MUCRAr}}{C^{MUCRA}})$; where C^{MUCRA} and C^{MUCRAr} are the costs associated to the optimal solutions found respectively by MUCRA and MUCRAr WDAs.

With the aim of assessing the most sensitive parameters with respect to *SI*, we employ a fractional factorial experiment design [4], assigning to each parameter different values.

We run 5756 instances of the experiments and for each run we sampled *SI*. In 150 cases (2.606%) the optimizer could not find an optimal solution within the time limit for MUCRA. In 289 cases (5.02%) the solver could not find an optimal solution within the time limit for MUCRAr. As explained above, the total number of samples that have been considered are $5756 - 150 = 5606$. Among these new samples, the optimizer could not find an optimal solution for MUCRAr for 191 (3.407%) tests.

We empirically observe that the savings of MUCRAr with respect to MUCRA go: (1) up to 44%; (2) beyond 3.29% in 50% of the cases; (3) beyond 8.59% in 30% of the cases. Next, we perform a sensitivity analysis in order to determine which parameters most affect an auction's outcome.

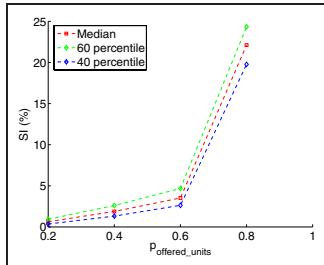


Figure 2. Variation of *SI* for different providers' capacities.

Sensitivity Analysis.

Figure 2 shows how the *SI* vary as the providers capacities increase. From it we can conclude that

C1: *The higher the providers' capacities, the higher the expected savings when introducing t-relationships.*

For this reason, when analysing the behavior of *SI* with respect to the remaining parameters, we will differentiate two cases: (1) $p_{offered_units} < 0.8$; and (2) $p_{offered_units} = 0.8$.

Figure 3 shows the variation of *SI* with respect to the number of required units. Our second conclusion is thus,

C2: *The finer the granularity of the transformations, the higher the expected savings when introducing t-relationships.*

The third factor significantly affecting *SI* is the relationship between the transformation costs of a buying agent with the providers' ones. Experimental results confirm that, as expected,

The behaviour of *SI* when changing the number of transformations within the TNS can be summarized by,

C4: *The more the number of transformations, the more the expected savings with respect to a MUCRA.*

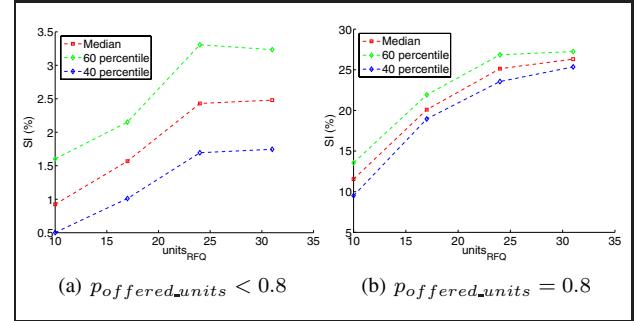


Figure 3. *SI* with respect to the number of required units.

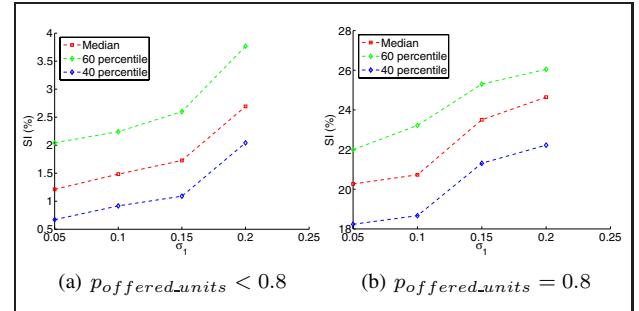


Figure 4. *SI* with respect to the variance of providers' prices.

Figure 4 shows the effect of varying the prices of goods among providers. The fifth conclusion follows:

C5: *The larger the market's prices spread, the higher the expected savings.*

To summarise, we can indeed confirm, based on the observation above, that there are market conditions (identified by **C1**, **C2**, **C3**, **C4**, and **C5**) wherein it is worth using MUCRAr instead of MUCRA.

3 Conclusions and Future Work

In this paper we have performed a set of experiments to quantitatively assess the potential savings in employing a MUCRAr instead of a MUCRA. Furthermore, we have also identified the market conditions for which MUCRAr is expected to lead to better auction outcomes to the auctioneer/buyer, namely: (1) markets with high-capacity providers; (2) auctions whose number of required units per good is large with respect to the units required by transformations (i.e. the likelihood of exploiting transformations is high); (3) auctions run by a buyer whose transformation (production) costs are cheaper than the providers' ones; and (4) markets where providers' competitiveness is not high (the more scattered the providers' competitiveness, the larger the expected savings).

REFERENCES

- [1] Andrea Giovannucci, Juan A. Rodriguez-Aguilar, and Jesús Cerdà, 'Multi-unit combinatorial reverse auctions with transformability relationships among goods', in *LNCS*, volume 3828, Hong Kong, China, (December 2005). www.iiia.csic.es/~andrea/papers/WINE318a.pdf.
- [2] T. Murata, 'Petri nets: Properties, analysis and applications', in *IEEE*, volume 77, pp. 541–580, (1989).
- [3] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, 'Winner determination in combinatorial auction generalizations', in *AAMAS 2002*, pp. 69–76, Bologna, Italy, (2002).
- [4] Box G.E.P. Hunter W.G. and Hunter J.S., *Statistics for experimenters An introduction to design, data analysis, and model building*, Wiley, 1978.

Count-as Conditionals, Classification and Context

Guido Boella¹ and Leendert van der Torre²

Abstract. Searle represents constitutive norms as count-as conditionals, written as ‘ X counts as Y in context C ’. Grossi *et al.* study a class of these conditionals as ‘in context C , X is classified as Y ’. In this paper we propose a generalization of this relation among count-as conditionals, classification and context, by defining a class of count-as conditionals as ‘ X in context C_0 is classified as Y in context C ’. We show that if context C_0 can be different from context C , then we can represent a larger class of examples, and we have a weaker logic of count-as conditionals.

1 Count-as conditionals, classification and context

Searle [7] argues that there is a distinction between two types of rules.

“Some rules regulate antecedently existing forms of behaviour. For example, the rules of polite table behaviour regulate eating, but eating exists independently of these rules. Some rules, on the other hand, do not merely regulate an antecedently existing activity called playing chess; they, as it were, create the possibility of or define that activity. The activity of playing chess is constituted by action in accordance with these rules. The institutions of marriage, money, and promising are like the institutions of baseball and chess in that they are systems of such constitutive rules or conventions” ([7], p. 131).

For Searle, regulative and constitutive norms are related via institutional facts like marriage, money and private property. They emerge from an independent ontology of “brute” physical facts through constitutive rules of the form “such and such an X counts as Y in context C ” where X is any object satisfying certain conditions and Y is a label that qualifies X as being something of an entirely new sort. E.g., “ X counts as a presiding official in a wedding ceremony”, “this bit of paper counts as a five euro bill” and “this piece of land counts as somebody’s private property”. Regulative norms refer to these institutional facts. E.g., consider a society which believes that a field fenced by an agent counts as the fact that the field is the agent’s property. The fence is a physical “brute” fact, while being a property is an institutional fact. Regulative norms forbidding trespassing refer to the abstract concept of property rather than to fenced fields.

Grossi *et al.* [4] study the relation between on the one hand count-as conditionals and on the other hand classification and context. They formalize a class of count-as conditionals as contextual classifications, and thus do not claim that all count-as conditionals can be represented in this way. Roughly, as we understand it, their idea of classification is that X and Y are interpreted as sets (of facts, objects,

events, actions, etc) and that ‘ X is classified as Y ’, or ‘ X is-a Y ’ for short, is interpreted as ‘the set of interpretations of X is a subset of the set of interpretations of Y ’. Thus, classification is the is-a relation frequently studied in conceptual modeling, for example as a subsumption relation in type theory, or as a T-Box expression in description logics.

Moreover, Grossi *et al.* use modal logic to represent their count-as conditionals as contextual classifications. They represent X and Y as propositions, and the context as a modal operator (to be precise, as a particular kind of KD45 modality). Roughly, representing the classification relation as a material implication ‘ $X \rightarrow Y$ ’, they propose the following definition.

‘ X counts as Y in context C ’ is represented by $[C](X \rightarrow Y)$.

For example, consider a regulative norm stating that vehicles are forbidden in the park, and the constitutive norm ‘bicycles count as vehicles in the park’. This count-as conditional classifies bicycles as vehicles in the context of being in the park, and can be formalized as $[park](bicycles \rightarrow vehicles)$.

The logic of Grossi *et al.* turns out to be much stronger than other logics of count-as conditionals, such as the one of Jones and Sergot [5]. In Jones and Sergot’s study of count-as conditionals, the logic of count-as conditionals is very weak. It just satisfies replacements of logical equivalents, left disjunction and right conjunction. Moreover, they are inclined to accept transitivity. In addition, the logic of Grossi *et al.* satisfies, for example, reflexivity and contraposition.

We believe that there are two important advantages in representing count-as conditionals as contextual classifications. The first advantage is that it may help to better understand constitutive norms. Defining count-as conditionals as contextual classifications might lead to a more precise characterization of count-as conditionals – though it will not cover the whole class of count-as conditionals.

The second advantage of defining count-as conditionals as contextual classifications is that it may help us to understand how count-as conditionals are related to regulative norms like obligations and permissions, which is one of the main open questions in normative systems. Since regulative norms can be defined as classifications of behaviors in obligatory, permitted and unnormed ones, count-as conditionals as contextual classifications may explain this relation.

Given these two advantages, we are interested in generalizing the class of count-as conditionals that can be considered as contextual classifications. One reason to look for generalizations is that the formalization does not seem to take Searle’s distinction between brute and institutional facts into account. For example, since X are brute facts and Y are institutional facts, this distinction may suggest that X and Y themselves refer to distinct contexts. Another reason is that it does not seem straightforward to represent examples. For example, what is the context in “this bit of paper counts as a five euro bill” or in “this piece of land counts as somebody’s private property”?

¹ Dipartimento di Informatica - Università di Torino - Italy. E-mail: guido@di.unito.it

² Department of Computer Science, University of Luxembourg. E-mail: leendert@vandertorre.com

2 Contexts as views

To generalize the notion of count-as conditionals as contextual classifications, we again make use of the terminology and methodology developed in conceptual modeling. As we already observed, classification is a central relation used in conceptual modeling. In software engineering, for example, the world (or a system) is modeled using various models of the Unified Modelling Language (UML) (class diagrams, sequence diagrams, *etc.*), and the is-a relation is used to define classification of concepts in class diagrams.

In this paper we use the notion of viewpoint and view to give an interpretation of context. A viewpoint is a particular way to look at the world [1]. For example, a structural viewpoint only describes the things staying the same over time, an action or process viewpoint describes the behavior over time, a brute viewpoint describes only brute facts, an institutional viewpoint describes only institutional facts, a power viewpoint describes the world in terms of powers of agents, a dependency viewpoint describes the world in terms of dependencies among actors, etc. Such a viewpoint is associated with a stakeholder having particular concerns; this aspect is not further discussed here. Each viewpoint gives a view on the system or the world.

There are two issues in conceptual modeling: defining each of these models or views, and defining the relations among them. In practice, it is the latter which is most problematic [3]. System descriptions often comprise many heterogeneous models and other descriptions, with ill-defined or completely lacking relations, inconsistencies, and a general lack of coherence and vision. UML for example does not define the relations among its models (nor does it define a semantics for its models).

In this interpretation, ' X counts as Y in context C ' is interpreted as ' X counts as Y in viewpoint C '. Our notion of context as view seems to be compatible with the framework of Grossi *et al.* In this interpretation, the conditional is interpreted as ' X is classified as Y in viewpoint C '. However, the count-as conditional can be used only to classify concepts within a single view, not to relate concepts in two or more views.

3 Count-as as relations among contexts or views

Generalizing the contextual classification to incorporate relations among contexts is straightforward. If we represent the classification relation again as a conditional ' $X \rightarrow Y$ ', then we propose the following definition.

' X in context C_0 counts as Y in context C ' is represented by
 $[C_0]X \rightarrow [C]Y$.

The following examples illustrate our definition. The third example illustrates how counts-as conditionals can be used to define properties of regulative norms too.

1. [brute](' p is a piece of paper') \rightarrow [institutional](' p is money'): This piece of paper p counts as money. The contexts or views are the physical world and social (institutional) reality, respectively.
2. [facts]('this is a piece of paper') \rightarrow [actions]('using the piece of paper for paying'): This piece of paper can be used to pay for things. The contexts or views are facts and actions, respectively.
3. [brute]('the occurrence of behavior p ') \rightarrow [deontic](' p is a violation'): Behavior p counts as a violation. The contexts or views are physical world and deontic reality, respectively.

The context or view C_0 often refers to the world of physical (observable) facts. As an abbreviation, we can thus use ' X counts as Y

in context C ' to mean that ' X in context 'brute facts' counts as Y in context C '. Moreover, as a first approximation of the relation between count-as conditionals, classification and context, we can use the following definition:

' X counts as Y in context C ' is represented by $(X \rightarrow [C]Y)$.

The advantage of this definition is that we can compare the logical properties among count-as conditionals of our proposal with the existing ones in the literature. In this case we end up with a weaker logic than the one proposed by Grossi *et al.*, since we no longer have reflexivity or contraposition. It is more in line with earlier proposals, such as the approach of Jones and Sergot.

4 Concluding remarks

Constitutive norms play an important role in normative systems like organizations and institutions, which have received much attention in artificial intelligence and multi-agent systems. According to Searle, the simple mechanism of count-as conditionals can explain the complexity of the construction of social reality. This aspect of constitutive norms does not get a satisfactorily explanation in Grossi *et al.*'s definition of count-as conditional as a contextual classification. It is hard to believe that all complex mechanisms found in social reality, can be explained by classifications only.

When count-as conditionals act also as bridges between contexts or views, then they can be used to define new contexts from existing ones. They can be used to relate for example observable brute facts and institutional facts, or observable and legal facts. This at least partly explains why constitutive norms play such an important role in the construction of social reality. However, there are still aspects of count-as conditionals which cannot be satisfactorily explained by our generalized contextual classifications. For example, as we have shown in [2], constitutive norms can also be used to formalize how normative systems like organizations can regulate their own evolution. How to incorporate this aspect is subject of further research.

Another subject of further research is the logical analysis of counts-as conditionals as contextual classifications. We do not have that X counts as X , the absence of this reflexivity property was suggested also by Jones and Sergot's analysis. We thus need a logical framework of conditionals that not necessarily satisfy this property, such as the framework of input/output logic [6].

Finally, a subject of further research is to examine relationships between concepts. For example, the context [facts] consists of the contexts [brute] and [institutional]. Existing logics of context may be useful here.

REFERENCES

- [1] IEEE Standards Association, 'IEEE 1471-2000', (2000).
- [2] G. Boella and L. van der Torre, 'Regulative and constitutive norms in normative multiagent systems', in *Proc. of 10th International Conference on the Principles of Knowledge Representation and Reasoning KR'04*, pp. 255–265, Menlo Park (CA), (2004). AAAI Press.
- [3] M. Lankhorst *et al*, *Enterprise architecture at work*, Springer Verlag, 2005.
- [4] D. Grossi, F. Dignum, and J.-J.Ch. Meyer, 'Contextual taxonomies', in *LNCS n. 3487: Proc. of CLIMA'04 Workshop*, pp. 33–51, Berlin, (2004). Springer Verlag.
- [5] A. Jones and M. Sergot, 'A formal characterisation of institutionalised power', *Journal of IGPL*, 3, 427–443, (1996).
- [6] D. Makinson and L. van der Torre, 'Input-output logics', *Journal of Philosophical Logic*, 29, 383–408, (2000).
- [7] J.R. Searle, *Speech Acts: an Essay in the Philosophy of Language*, Cambridge University Press, Cambridge (UK), 1969.

Fair Distribution of Collective Obligations

Guido Boella¹ and Leendert van der Torre²

Abstract. In social mechanism design, obligation distribution creates individual or contractual obligations that imply a collective obligation. A distinguishing feature from group planning is that also the sanction of the collective obligation has to be distributed, for example by creating sanctions for the individual or contractual obligations. In this paper we address fairness in obligation distribution for more or less powerful agents, in the sense that some agents can perform more or less actions than others. Based on this power to perform actions, we characterize a trade-off in negotiation power. On the one hand, more powerful agents may have a disadvantage during the negotiation, as they may be one of the few or even the only agent who can see to some of the actions that have to be performed to fulfill the collective obligation. On the other hand, powerful agents may have an advantage in some negotiation protocols, as they have a larger variety of proposals to choose from. Moreover, powerful agents have an advantage because they can choose from a larger set of possible coalitions. We present an ontology and measures to find a fair trade-off between these two forces in social mechanism design.

1 Ontology

Power may affect the distribution of obligation in various ways, and we therefore propose to analyze the obligation distribution problem in terms of social concepts like power and dependence. Power has been identified as a central concept for modeling social phenomena in multi-agent systems by various authors [5, 6, 8, 12], as Castelfranchi observes both to enrich agent theory and to develop experimental, conceptual and theoretical new instruments for the social sciences [6]. In most of these proposals, power is interpreted as the ability of agents to achieve goals. For example, in the so-called power view on multi-agent systems [2], a multi-agent system consists of a set of agents (A), a set of goals (G), a function that associates with each agent the goals the agent desires to achieve (*goals*), and a function that associates with each agent the sets of goals it can achieve (*power*). To be precise, to represent conflicts the function *goals* returns a set of set of goals for each set of agents. Such abstract structures have been studied as qualitative games by Wooldridge and Dunne [11].

To model the role of power in obligation distribution in normative multiagent systems, we associate with each norm the set of agents that has to fulfill it, and of each norm we represent how to fulfill it, and what happens when it is not fulfilled.

- First, we associate with each norm n a set of goals $O(n) \subseteq G$. Achieving these normative goals $O(n)$ means that the norm n has been fulfilled; not achieving these goals means that the norm is

violated. We assume that every normative goal can be achieved by the group, i.e., that the group has the power to achieve it.

- Second, we associate with each norm a set of goals $V(n)$ which will not be achieved if the norm is violated (i.e., when its goals are not achieved), this is the sanction associated to the norm. We assume that the sanction affects at least one goal of each agent of the group the obligation belongs to, and that the group of agents does not have the power to achieve these goals.

Definition 1 Let a normative multi-agent system be a tuple $\langle A, G, \text{goals}, \text{power}, N, OD, O, V \rangle$ where:

- the agents A , goals G and norms N are three finite disjoint sets;
- $\text{goals} : A \rightarrow 2^G$ is a function that associates with each agent the goals the agent desires;
- $\text{power} : 2^A \rightarrow 2^{2^G}$ is a function that associates with each set of agents the sets of goals the set of agents can achieve;
- $OD : N \rightarrow 2^A$ is a function that associates with every norm a set of agents that have to see to it that the norm is fulfilled.
- $O : N \rightarrow 2^G$ is a function that associates with each norm the goals which must be achieved to fulfill the norm; We assume for all $n \in N$ that there exists $H \in \text{power}(OD(n))$ with $O(n) \subseteq H$;
- $V : N \rightarrow 2^G$ is a function that associates with each norm the goals that will not be achieved if the norm is violated; We assume that for all $n \in N$ and $a \in OD(n)$ that $V(n) \cap \text{goals}(a) \neq \emptyset$, and for all $B \subseteq OD(n)$ and $H \in \text{power}(B)$ that $V(n) \cap H = \emptyset$.

2 Power of the people

Some studies of obligation distribution consider logical relations among collective and individual obligations [9]. Cholvy and Garion [7] claim that if there is a set of agents subject to an obligation to perform some task, then the derivation of individual obligations from collective obligations depends on several parameters, among which the ability of the agents. If an agent is the only one able to perform a part of that task, then it is obliged to do that part and it is also obliged to do that towards the other members of the set.

For example, they consider three children who are obliged by their mother to prepare the table for dinner. The oldest child is the only one who is tall enough to get the glasses on the cupboard. The whole group is responsible for the violation of the collective obligation, but in case the violation is due to the fact that the oldest boy did not bring the glasses, only he can be taken responsible by the group because he was the only one able to take the glasses.

This disadvantage of more powerful agents is measured for each agent a and norm n as the number of normative goals it has the power to achieve, each weighted by the number of other agents within the group that can achieve the same goal. $m_1(a, n) = 0$ if $a \notin OD(n)$, and otherwise:

$$\bullet m_1(a, n) = \sum_{g \in \text{power}(a) \cap O(n)} \frac{1}{|\{b \in OD(n) | g \in \text{power}(b)\}|}$$

¹ Dipartimento di Informatica - Università di Torino - Italy. E-mail: guido@di.unito.it

² Department of Computer Science, University of Luxembourg. E-mail: leendert@vandertorre.com

3 Power of the king

An analysis based on power and dependence, as is common in social theory, suggests that the ability possessed by only one agent makes the remaining agents depend on him, since they lack the power to do part of the task they are obliged to. In this sense the oldest boy is in the best position, rather than having an additional burden and being sanctioned both for not respecting the collective obligation and his own obligation.

The dependence on more powerful agents can be made explicit when the negotiation process is made explicit. For example, the oldest boy has more power in the negotiation for the distribution of the task, and, by exercising this power, he may end up doing less than the other boys. An agreement is the result of a negotiation which has to take into account the dependence relations among the agents. E.g., the kids are first obliged to negotiate the distribution of the obligation, and then, only if they find a successful distribution of obligations, the oldest kid becomes obliged to see to the glasses. Note that in this negotiation model the distribution is not centralized, which is another reason why it is more complex than a planning problem.

In the negotiation protocol proposed in [4], each agent in turn makes a proposal of a complete distribution, which can be accepted or rejected by the other agents. Agents may not repeat their own proposal. If no proposal is accepted, then the norm will be violated. Roughly, when an agent can make many proposals, he has the advantage that he can propose deals which are beneficial to himself in all rounds. This models the intuition that an agent with more possible proposals has more power in the negotiation. This holds regardless of the penalty for breaking the negotiation.

For example, assume for a norm n that an agent $a \in OD(n)$ is allowed to propose a distribution $d(n) : OD(n) \rightarrow 2^{\text{power}(OD(n))}$ that associates with every agent some of the goals he has the power to achieve, such that the distribution implies all the goals of the norm $O(n) \subseteq \cup_{a \in OD(n)} d(n)(a)$ (note that here we assume that if a goal can be achieved by a set of agents, then it can be achieved by an individual agent). With $D(n)$ we refer to the set of all such $d(n)$. Moreover, assume that an agent cannot make two proposals in which itself is dealt the same goals. The measure of this example is the number of elements of $D(n)$ in which $d(n)(a)$ is distinct.

$$\bullet m_2(a, n) = |\{d(n)(a) \mid d(n) \in D(n)\}|$$

If the distributions that may be proposed have to satisfy other criteria, for example related to the distribution of the sanction [4], then the definition of d can be adapted accordingly.

4 Coalition power

Agents depend on more powerful agents not only to accomplish collective obligations, but also to fulfill their other desires. Thus, if we not only consider the collective obligation that has to be distributed but also the other desires of the agents, then powerful agents have an additional advantage in negotiation. For example, they may threaten to do something the other agents dislike, who therefore will accept an inferior solution.

A framework to study such dependencies is coalition formation. In contrast to the negotiation model in the previous section, coalition formation considers not only a single collective obligation to be distributed, but many of them. Such a setting is given in our definition of normative multi-agent system in Definition 1. A way to calculate the possible coalitions from a power view on normative multi-agent systems is given in [1, 3]. In that paper, a coalition is only considered

in a power structure if each agent contributes and receives something in a coalition (called do-ut des).

$$\bullet m_3(a) = |\text{coalitions}(a)|$$

Of the three measures proposed in this paper, this third measure is the most abstract. It is tempting to make it more precise by quantifying over the goals of the norms the agent is involved in. However, the impact of the dependencies in coalition formation on obligation distribution has not been studied in any detail thus far, and therefore we have chosen this more general measure. Consequently, it can only be taken as an indication of the impact on obligation distribution.

5 Concluding remarks

In some social mechanisms the agents with more power will end up doing more, but in other ones they will end up doing less. In the extreme case, a very powerful agent might do nothing at all - or the distribution is not made at all. The measures discussed in this paper can be used to enforce fairness in the distribution of agents. Another solution is to build a normative system that ensures a fair distribution. In this approach, there are norms that, for example, indicate the number of deals an agent can propose.

Most multi-agent system models concerned with complex cognitive tasks like obligation creation, negotiation and distribution [4, 7, 9, 10] are based on relatively detailed cognitive models incorporating, for example, beliefs, obligations, desires, intentions, goals, preferences, and so on. Some of these elements can be used to make the measures more precise. For example, another advantage may have more reliable knowledge about the goals and abilities of other agents. If we extend the model with such beliefs of agents about the power and goals of other agents, then this advantage can be measured too.

REFERENCES

- [1] G. Boella, L. Sauro, and L. van der Torre. An abstraction from power to coalition structures. In *Proceedings of ECAI 2004*, pages 965–966, 2004.
- [2] G. Boella, L. Sauro, and L. van der Torre. Social viewpoints on multiagent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, pages 1358–1359, 2004.
- [3] G. Boella, L. Sauro, and L. van der Torre. Strengthening admissible coalitions. In *Proceedings of ECAI 2006*, 2006.
- [4] G. Boella and L. van der Torre. Negotiating the distribution of obligations with sanctions among autonomous agents. In *Procs. of ECAI 2004*, pages 13–17, 2004.
- [5] S. Brainov and T. Sandholm. Power, dependence and stability in multiagent plans. In *Procs. of AAAI99*, pages 11–16, 1999.
- [6] C. Castelfranchi. Micro-macro constitution of power. *ProtoSociology*, 18–19, 2003.
- [7] L. Cholvy and C. Garion. Collective obligations, commitments and individual obligations: a preliminary study. In *Procs. of ΔEON02*, London, 2002.
- [8] R. Conte and J. Sichman. Multi-agent dependence by dependence graphs. In *Procs of AAMAS02*, pages 483–490, 2002.
- [9] F. Dignum and L. Royakkers. Collective obligation and commitment. In *Procs. of 5th Int. Conference on Law in the Information Society*, Florence, 1998.
- [10] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: an organizational view of multiagent systems. In *Procs. of AOSE03*, LNCS 2935, page 214230. Springer, 2003.
- [11] M. Wooldridge and P.E. Dunne. On the computational complexity of qualitative coalitional games. *Artif. Intell.*, 158(1):27–73, 2004.
- [12] Lopez y Lopez. *Social Power and Norms: Impact on agent behaviour*. PhD thesis, June 2003.

Proactive Identification of a Physician's Information Needs¹

Loes M.M. Braun² and Floris Wiesman³ and H. Jaap van den Herik² and Arie Hasman³

1 INTRODUCTION

Information retrieval (IR) in the medical domain has proven to be vital to the quality of care [3]. However, physicians often lack the time and skills to formulate adequate information needs and in some situations physicians are even not aware of their information needs [2]. We define an *information need* as an expression of missing information needed to perform a particular task – in this case treating a patient. An example of an information need is *Does Clarithromycin cause pancreatitis?*

Our goal is to investigate to which extent an agent can support a physician in formulating his⁴ information needs, without requiring any effort from the physician. Therefore, we have developed and implemented an intelligent agent that identifies patient-related information needs automatically and proactively. Based on these information needs, the agent can retrieve relevant, patient-related literature.

For this purpose, we have modelled general information needs into information-need templates [1]. We consider an *information-need template* to be an abstract, natural-language representation of an information need. For instance, the information-need template corresponding to the information need *Does Clarithromycin cause pancreatitis?* is *Does [Chemical] cause [Disease or Syndrome]?* An information-need template contains data slots that can be filled with medical concepts.

Our agent transforms information-need templates into patient-related information needs by instantiating the data slots in the templates with data from the electronic patient record (EPR) of a specific patient [1]. The resulting information needs are used to initiate search actions in medical literature databases without requiring any effort of the physician.

Experiments indicate that the number of information needs formulated by our agent is too high for adequate use [1]. Such a high number of information needs will lead to a *literature overload* of the physician, i.e., the retrieval of an unmanageable amount of literature. Therefore, our agent should be improved, i.e., the number of formulated information needs should be reduced. That is the topic of this two-pager.

2 REDUCING INFORMATION NEEDS

One way to reduce the number of information needs is by employing additional knowledge. We focus on six types of knowledge that

¹ This research is funded by the Netherlands Organisation for Scientific Research (grant number 634.000.021).

² MICC-IKAT, Maastricht University, email: {l.braun; herik}@cs.unimaas.nl

³ AMC, University of Amsterdam, email: {f.j.wiesman; a.hasman}@amc.uva.nl

⁴ For brevity we will use the pronoun he (his) where he or she (his or her) is meant.

might be used by our agent, viz. (a) entry time, (b) entry order, (c) domain knowledge, (d) specialism, (e) user model, and (f) formulation history.

Entry time. The entry time represents the time at which data are entered into the EPR. Our agent may use knowledge of this type to determine the relevance of specific data entries. If data are entered outside a certain time interval, they might be considered irrelevant and can be discarded during the formulation of information needs.

Entry order. The entry order represents the order in which data are entered into the EPR. Our agent can use this type of knowledge to detect nonsensical information needs. The entry order is particularly applicable to information-need templates containing data slots that are temporally dependent.

Domain knowledge. Many hospitals use alert systems that automatically forewarn physicians about potentially harmful actions. The alerts issued by an alert system fulfil some of the physician's information needs. Therefore, our agent should discard these information needs. To determine which information needs should be discarded, our agent has to predict the alerts by using the domain knowledge employed by the alert system.

Specialism. In a multidisciplinary hospital department, a physician's information needs are probably closely connected to his specialism. Consequently, our agent might discard information needs if they are not connected to the physician's specialism.

User model. By employing a user model, the information preferences of a physician are recorded, actively or passively. When preferences are recorded actively, the physician indicates what information he prefers. When preferences are recorded passively, the system monitors the use of the system by the physician and infers the physician's preferences from the recorded data. Since a physician's preferences may change over time, the information model can be adapted at any given moment.

Formulation history. A physician's formulation history represents the information needs that have been formulated previously. This knowledge can be used to prevent the agent from formulating information needs repeatedly. Information needs that have been formulated for a physician should not be formulated anew for that physician within a certain time frame.

The usefulness of each of the knowledge types presented above depends heavily on the specific cause underlying the excess of information needs. To determine the nature of this cause, we performed an experimental analysis of the information needs formulated. The results of the analysis provide directions for determining which knowledge type is most appropriate for our purpose.

In the experiment, we simulated the formulation of information needs based on patient data in the field of intensive care medicine.

The patient data concerned 82 patients and were entered by 31 physicians over six months of time. The patient data included data entries of four different types, viz. (a) disease or syndrome, (b) diagnostic procedure, (c) therapeutic or preventive procedure, and (d) chemical. We confined the analysis to these types, since they represent the majority of the data entries in the EPRs. The information-need templates used concerned three subjects, viz. (a) prognosis, (b) therapy (including prevention), and (c) side effects (including etiology).

The results of our experiment indicated that five of the information-need templates (see table 1) generated many more information needs than the other templates. From our analysis of these five templates we may conclude that the large number of information needs can only be explained by the large amount of patient data available. Consequently, reducing the number of data entries that are used to instantiate information-need templates seems the best way to reduce the number of information needs formulated.

Based on the observations above, we have ordered the six knowledge types according to their ability to reduce the number of patient data (see conclusions).

3 EXPERIMENTS AND RESULTS

To determine the actual performance of the knowledge types presented in section 2, we conducted an experiment. We limited the scope of the experiment in two respects: (a) we focused on the two knowledge types that are most appropriate according to our analysis (section 2): formulation history and entry order, and (b) we focused on the five information-need templates generating the majority of information needs (table 1).

To determine the performance of the knowledge types we let our agent formulate information needs in four different settings: (i) using no additional knowledge, (ii) using only the formulation history, (iii) using only the entry order, and (iv) using both knowledge types simultaneously.

Figure 1 shows the results of our experiment. With respect to figure 1, three observations can be made.

First, regarding the formulation history, we observe that for all five templates, the use of formulation history results in a reduction. The largest reduction (75%) occurs in template (c). Furthermore, for four of the templates (a, c, d, and e) the use of the formulation history realizes a larger (or equal) reduction than the use of the entry order, as was expected (see section 2).

Second, with respect to the entry order, we observe that for four of the templates (a, b, c, and d) the use of the entry order results in a reduction. The largest reduction (75%) occurs in template (c). For template (e) the use of the entry order does not result in a reduction. For template (b), the use of the entry order results in a larger reduc-

Table 1. Information-need templates generating most information needs.

- (a) Can [Chemical] be used as a prophylaxis against [Disease or Syndrome] in a patient with [Disease or Syndrome] and no clinical risk factors for [Disease or Syndrome]?
- (b) Give a description of the advantages and disadvantages of [Health Care Activity] in relation to [Health Care Activity] in case of [Finding]/[Disease or Syndrome].
- (c) How can [Health Care Activity] be used on the evaluation of [Health Care Activity]?
- (d) How does [Health Care Activity] compare with [Health Care Activity] in the setting of [Sign or Symptom]/[Pathologic Function]?
- (e) What is the interaction of [Chemical] with [Chemical]?

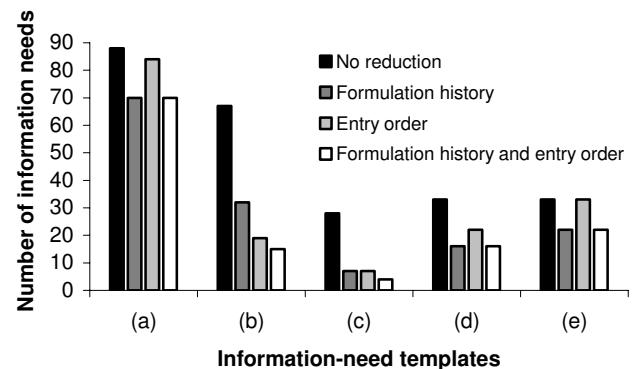


Figure 1. Number of information needs formulated from five information-need templates, using various knowledge types.

tion than the use of the formulation history. Furthermore, for all five templates, the use of the entry order results in a smaller reduction than the use of both knowledge types simultaneously.

Third, regarding the use of both knowledge types simultaneously, we observe that for two templates (b) and (c) the combination of both knowledge types realizes a greater reduction than the use of each knowledge type separately. The largest reduction (85.7%) occurs in template (c). For the remaining three templates, the reduction resulting from the combination of both knowledge types is equal to the reduction resulting from the use of the formulation history only.

4 CONCLUSIONS

In this paper we presented six types of knowledge that may be used to reduce the number of information needs formulated by our agent. Furthermore, we ordered the knowledge types according to their usefulness in the field of intensive care medicine and assessed the performance of the two most-promising knowledge types. With respect to these investigations we arrived at three conclusions.

First, we may conclude that the large number of information needs formulated is mainly due to the large number of data entries used to instantiate information-need templates.

Second, we tentatively conclude that the knowledge types can be ordered according to their appropriateness as follows: (1) formulation history, (2) entry order, (3) specialism, (4) domain knowledge, (5) entry time, and (6) user model.

Third, we provisionally conclude that for four of the templates in table 1, the use of the formulation history results in a reduction larger than or equal to the use of the entry order, as was expected.

Finally, we remark that for two of the templates in table 1, combining the formulation history and the entry order has a positive effect.

REFERENCES

- [1] L.M.M. Braun, F. Wiesman, H.J. van den Herik, A. Hasman, and H.H.M. Korsten, 'Towards automatic formulation of a physician's information needs', *Journal of Digital Information Management*, 3(1), 40–47, (2005).
- [2] J.W. Ely, J.A. Osheroff, M.H. Ebell, M.L. Chambliss, D.C. Vinson, J.J. Stevermer, and E.A. Pifer, 'Obstacles to answering doctor's questions about patient care with evidence: Qualitative study', *British Medical Journal*, 324(7339), 710–716, (2002).
- [3] S. Gamble, 'Hospital libraries enhance patient care and save money', *Journal of the Alberta Association of Library Technicians*, 23(2), 10–12, (1996).

Acyclic Argumentation: Attack = Conflict + Preference

Souhila Kaci¹ and Leendert van der Torre² and Emil Weydert²

Abstract. In this paper we study the fragment of Dung's argumentation theory in which the strict attack relation is acyclic. We show that every attack relation satisfying a particular property can be represented by a symmetric conflict relation and a transitive preference relation in the following way. We define an instance of Dung's abstract argumentation theory, in which 'argument A attacks argument B' is defined as 'argument A conflicts with argument B' and 'argument A is at least as preferred as argument B', where the conflict relation is symmetric and the preference relation is transitive. We show that this new preference-based argumentation theory characterizes the acyclic strict attack relation, in the sense that every attack relation defined as such a combination satisfies the property, and for every attack relation satisfying the property we can find a symmetric conflict relation and a transitive preference relation satisfying the equation.

1 Acyclic argumentation framework

Argumentation is a reasoning model based on constructing arguments, determining potential conflicts between arguments and determining acceptable arguments. Dung's framework [7] is based on a binary attack relation among arguments. We restrict ourselves to *finite* argumentation frameworks, i.e., when the set of arguments \mathcal{A} is *finite*.

Definition 1 (Argumentation framework) An argumentation framework is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$ where \mathcal{A} is a set of arguments and \mathcal{R} is a binary attack relation defined on $\mathcal{A} \times \mathcal{A}$.

An acyclic argumentation framework is an argumentation framework in which the attack relation is acyclic, a symmetric argumentation framework is an argumentation framework in which the attack relation is symmetric, etc. In this paper we define an acyclic strict attack relation as follows. Assume the attack relation is such that there is an attack path where argument A_1 attacks argument A_2 , argument A_2 attacks argument A_3 , etc, and argument A_n attacks argument A_1 , then we have that all the arguments in the attack path attack the previous one. Consequently, if argument A strictly attacks B if A attacks B and not vice versa, then the strict attack relation is acyclic.

Definition 2 (Acyclic argumentation framework) A strictly acyclic argumentation framework is an argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ in which the attack relation $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ satisfies the following property:

If there is a set of attacks $A_1 \mathcal{R} A_2, A_2 \mathcal{R} A_3, \dots, A_n \mathcal{R} A_1$ then we have that $A_2 \mathcal{R} A_1, A_3 \mathcal{R} A_2, \dots, A_1 \mathcal{R} A_n$.

The semantics of Dung's argumentation framework are based on the two notions of defence and conflict free.

¹ C.R.I.L., Rue de l'Université SP 16 62307, Lens, France.

² Department of Computer Science, University of Luxembourg, Luxembourg.

Definition 3 (Defence) A set of arguments \mathcal{S} defends A iff for each argument B of \mathcal{A} which attacks A , there is an argument C in \mathcal{S} which attacks B .

Definition 4 (Conflict-free) Let $\mathcal{S} \subseteq \mathcal{A}$. The set \mathcal{S} is conflict-free iff there are no $A, B \in \mathcal{S}$ such that $A \mathcal{R} B$.

The following definition summarizes various semantics of acceptable arguments proposed in the literature. The output of the argumentation framework is derived from the set of selected acceptable arguments with respect to an acceptability semantics.

Definition 5 (Acceptability semantics) Let $\mathcal{S} \subseteq \mathcal{A}$.

- \mathcal{S} is admissible iff it is conflict-free and defends all its elements.
- A conflict-free \mathcal{S} is a complete extension iff we have $\mathcal{S} = \{A \mid \mathcal{S} \text{ defends } A\}$.
- \mathcal{S} is a grounded extension iff it is the smallest (for set inclusion) complete extension.
- \mathcal{S} is a preferred extension iff it is a maximal (for set inclusion) complete extension.
- \mathcal{S} is a stable extension iff it is a preferred extension that attacks all arguments in $\mathcal{A} \setminus \mathcal{S}$.

For the general case, in which the attack relation can be any relation, many properties and relations among these semantics have been studied. However, of instances of Dung's argumentation framework, not much is known (with the exception of symmetric argumentation frameworks, see [6]). In this paper we consider acyclic argumentation.

2 Conflict+preference argumentation framework

Consider an argumentation theory in which each argument is represented by a propositional formula, and 'argument A attacks argument B' is defined as ' $A \wedge B$ is inconsistent'. Unfortunately, such a simple argumentation theory is not very useful, since the attack relation is symmetric, and the various semantics reduce to, roughly, one of the following two statements: 'an argument is acceptable iff it is part of all/some maximal consistent subsets of the set of arguments.' However, such a simple argumentation theory is useful again when we add the additional condition that argument A is at least as preferred as argument B. We start with some definitions concerning preferences.

Definition 6 A pre-order on a set \mathcal{A} , denoted \succeq , is a reflexive and transitive relation. \succeq is total if it is complete and it is partial if it is not. The notation $A_1 \succeq A_2$ stands for A_1 is at least as preferred as A_2 . \succ denotes the order associated with \succeq . $A_1 \succ A_2$ means that we have $A_1 \succeq A_2$ without $A_2 \succeq A_1$.

The new preference-based argumentation framework considers a conflict and a preference relation. The conflict relation should not be interpreted as an attack relation, since a conflict relation is symmetric, and an attack relation is usually asymmetric.

Definition 7 (Conflict+preference argumentation framework) A conflict+preference argumentation framework is a triplet $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ where \mathcal{A} is a set of arguments, \mathcal{C} is a binary symmetric conflict relation defined on $\mathcal{A} \times \mathcal{A}$ and \succeq is a (total or partial) pre-order (preference relation) defined on $\mathcal{A} \times \mathcal{A}$.

Starting with a set of arguments, a symmetric conflict relation, and a preference relation, we combine this conflict relation with preference relation to compute Dung's attack relation. Then we use any of Dung's semantics to define the acceptable set of arguments. In contrast to most other approaches [1, 8] (but see [2, 3] for exceptions), our approach to reason about preferences in argumentation does not refer to the internal structure of the arguments.

Definition 8 Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ a conflict+preference argumentation framework. We say that $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ represents $\langle \mathcal{A}, \mathcal{R} \rangle$ iff for all arguments A and B of \mathcal{A} , we have $A \mathcal{R} B$ iff $A \mathcal{C} B$ and $A \succeq B$. We also say that \mathcal{R} is represented by \mathcal{C} and \succeq .

Since the attack relation is defined on the basis of conflict \mathcal{C} and preference relation \succeq , also the other relations defined by Dung are reused by the conflict+preference argumentation framework. For example, to compute the grounded semantics of the conflict+preference framework, first compute the attack relation, and then compute the grounded semantics as in the general case.

3 Acyclic attack = conflict + preference

To prove that acyclic attacks are characterized by conflicts and preferences, we have to show that the implication holds in both ways. We first show that the implication from right to left holds, which is the easiest direction.

Lemma 1 If the conflict+preference argumentation framework $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ represents the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$, then $\langle \mathcal{A}, \mathcal{R} \rangle$ is a strictly acyclic argumentation framework (in the sense of Definition 2).

Proof. Assume a set of attacks $A_1 \mathcal{R} A_2, A_2 \mathcal{R} A_3, \dots, A_{n-1} \mathcal{R} A_n, A_n \mathcal{R} A_1$. Since $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ represents $\langle \mathcal{A}, \mathcal{R} \rangle$, we have also $A_i \mathcal{C} A_{(i \bmod n)+1}$ and $A_i \succeq A_{(i \bmod n)+1}$. Due to symmetry of \mathcal{C} , we also have $A_{(i \bmod n)+1} \mathcal{C} A_i$. Moreover, due to transitivity of \succeq , we have $A_{(i \bmod n)+1} \succeq A_i$ for $1 \leq i \leq n$. Consequently we have $A_{(i \bmod n)+1} \mathcal{R} A_i$, and thus $\langle \mathcal{A}, \mathcal{R} \rangle$ is strictly acyclic.

Now we show that the implication from left to right holds. We prove this lemma by construction: given an acyclic argumentation framework, we construct a conflict+preference framework representing it.

Lemma 2 If $\langle \mathcal{A}, \mathcal{R} \rangle$ is a strictly acyclic argumentation framework, then there is a conflict+preference argumentation framework $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ that represents it.

Proof. Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be a strictly acyclic argumentation framework. Moreover, consider a conflict+preference argumentation framework $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ defined as follows:

- $\mathcal{C} = \{(a, b) \mid (a, b) \in \mathcal{R} \text{ or } (b, a) \in \mathcal{R}\}$ is the symmetric closure of \mathcal{R}
- \succeq is the transitive and reflexive closure of \mathcal{R}

$\mathcal{R} \subseteq \mathcal{C} \cap \succeq$ by construction. The other direction we prove by contradiction, so assume ACB and $A \succeq B$ without ARB . Since ACB we have also BCA (since \mathcal{C} is symmetric) and we must have BRA (due to ACB and BCA we have either ARB or BRA). Since $A \succeq B$ we must have either $A = B$ (added by reflexive closure) or there is an attack path from A to B (added by transitivity). In both cases we obtain a contradiction:

1. If $A = B$ then BRA without ARB is directly a contradiction.
2. If there is a path from A to B , then together with BRA there is a cycle, and due to the acyclicity property we have ARB , which contradicts the assumption.

Summarizing, strictly acyclic argumentation frameworks are characterized by conflict+preference argumentation frameworks.

Theorem 1 $\langle \mathcal{A}, \mathcal{R} \rangle$ is a strictly acyclic argumentation framework if and only if there is a conflict+preference argumentation framework $\langle \mathcal{A}, \mathcal{C}, \succeq \rangle$ that represents it.

4 Concluding remarks

The preference-based argumentation theory introduced in this paper is a variant of the preference based framework of Amgoud and Cayrol [1], who define that argument A attacks argument B if A defeats B and B is not preferred to A . Note that our representation theorem does not hold for such a definition. As far as we know it is an open problem which properties the attack relation satisfies (if the defeat relation is symmetric).

Another subject for further research is how to use the strictly acyclic or conflict+preference frameworks. It may also be worthwhile to consider the generalizations of Dung's framework in propositional argumentation [4, 5].

Finally, a subject for further study is the complexity of the argumentation frameworks discussed in this paper. Is the computation of acceptable arguments in some sense easier or more efficient for strictly acyclic argumentation frameworks? Also, can we find more efficient or anytime algorithms for these frameworks?

REFERENCES

- [1] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *AMAI Journal*, 34:197–216, 2002.
- [2] L. Amgoud, S. Parsons, and L. Perrussel. An argumentation framework based on contextual preferences. Technical report, Department of Electronic Engineering, Queen Mary and Westfield College, 2000.
- [3] T. Bench-Capon. Persuasion in practical argument using value based argumentation framework. *Journal of Logic and Computation*, 13(3):429–448, 2003.
- [4] A. Bochman. Propositional Argumentation and Causal Reasoning. In *11th Int. Joint Conf. on Artificial Intelligence*, pages 388–393, 2005.
- [5] G. Boella, J. Hulstijn, and L. van der Torre. A logic of abstract argumentation. In *Postproceedings of ArgMAS05*, LNCS. Springer, 2006.
- [6] S. Coste-Marquis, C. Devred, and P. Marquis. Symmetric argumentation frameworks. In *Proceedings of ECSQARU-2005*, pages 317–328, 2005.
- [7] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n -person games. *Artificial Intelligence*, 77:321–357, 1995.
- [8] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.

Semantic Knowledge Model and Architecture for Agents in Discrete Environments

Michal Laclavik and Marian Babik and Zoltan Balogh and Emil Gatial and Ladislav Hluchy¹

1 Introduction

Multi-Agent Systems (MAS) is a powerful paradigm for distributed heterogeneous information systems when representation and reasoning using knowledge is needed. According to the Agent Technology Roadmap [3], which is the result of AgentLink, there are a number of broad technological challenges for research and development over the next decade in the agent technology. Within the presented work we are trying to partially cover some of them. These are:

- Providing better semantic infrastructure
- Apply basic principles of software and knowledge engineering
- Make stronger connection with existing commercial technologies

In the MAS, knowledge is usually represented by states, rules or predicate logic [12] [2]. Although this is extremely powerful, it is hard to capture knowledge from a person or from current information systems in the form of states or predicate logic clauses. Moreover, another difficulty is to present information and knowledge expressed in e.g. predicate logic to the end user [8]. Ontology as understood in the Semantic Web is closer to current information systems. It is based on XML/RDF [9], which can be more easily captured or returned from/to a person and existing information systems because information systems usually have XML based interfaces and XML can be easier presented to user after XSL transformations [10].

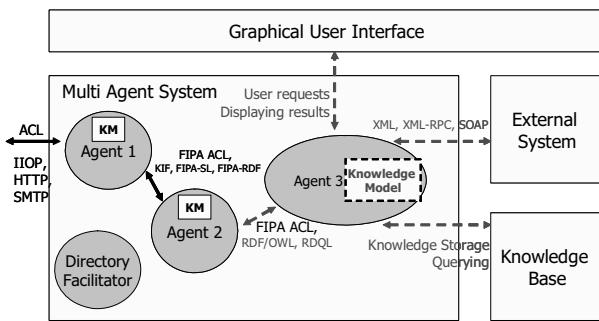


Figure 1. Missing features of FIPA compliant agent architecture

The current MAS platforms follows FIPA [12] standards. FIPA [12] describes the knowledge model based on ontology, but leaves internal agent memory model, its manipulation and understanding of design of agent on developer's decisions. In addition FIPA defines a

¹ Institute of Informatics, Slovak Academy of Sciences, Dubravská cesta 9, 845 07 Bratislava, Slovakia, email: laclavik.ui@savba.sk

knowledge manipulation based on content languages such as FIPA-SL [12], FIPA-KIF, which are powerful but lack any interconnection with commercial tools and standards. JADE agent system for example supports FIPA-SL language for message passing, but no FIPA-SL query engine or repository of such knowledge model is available. This is why we see ontology description from the semantic web area (RDF/OWL [9]) more appropriate for real application. Therefore we have decided to integrate semantic web technologies into MAS and create architecture, methodology and software for such integration. In addition, we have developed a generic ontology suitable for representing an agent knowledge model suitable for discrete environment. This model can be further extended, thus enabling its use in many areas especially for the knowledge and experience management [14], [6], [8].

2 Agent Knowledge Model

Most of the agent architectures are combinations of basic architecture types - the so called hybrid architectures. We focused on behavioral architecture, where an agent memory model used for behaviors implementations was created within this work. Proposed Knowledge Model is generic and suitable for applications with discrete, fully observable environments where environment changes can be captured by events.

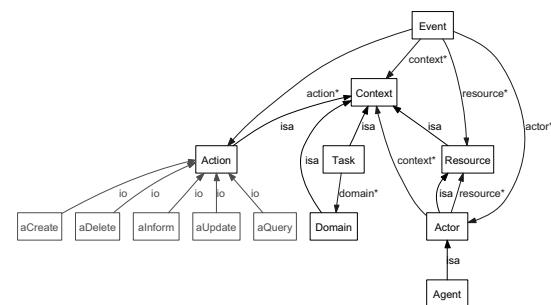


Figure 2. Basic Ontology for Knowledge Modeling

Our model is based on five main elements: Resources, Actions, Actors, Contexts and Events. Figure 2 shows formal graph representation of a model. The proposed model is described also using Description Logic [7] compatible with OWL-DL [9]. In proposed model we expect that all agents share same ontology.

Here we describe only the *Actor* class using Description Logic, which consists of important properties: *context.Actor*, *resource.Actor*.

$$\begin{aligned} Actor \subseteq \\ resource.Actor(\{resource\}) \cap \\ context.Actor(\{actor\}) \cap \\ Resource \end{aligned} \quad (1)$$

The *context.Actor* represents actor's current context. The system or application environment is based on stored events. The events model the environment state. Current state of the environment or actor related environment/context is thus affected by relevant new events.

$$\begin{aligned} context.Actor(\{actor\}) = \\ f_C(\forall event; actor.Event(\{actor\}) \in \{event\}) \end{aligned} \quad (2)$$

The *resource.Actor* property stands for all current resources of the actor. These resources are results of actors' intentions or objectives. Such resources are dependable on current actors' environment state/context (*context.Actor*).

$$\begin{aligned} resource.Actor(resource) = \\ f_R(contextActor(\{actor\})) \end{aligned} \quad (3)$$

Functions/algorithms for context and resource updating are specified by (2) and (3). An advantage of such model is that it enables achievement of better results when such algorithms are changed in the future, using the same model and data. Due to storing all events we can model the environment at any moment from past and process it later from any starting point with improved algorithms for context and resource updating. In addition, this model can be successfully used outside the MAS [6]. It can be used in the knowledge intensive application, where we need to model actors and their knowledge model. Often this is the case when we need to model users of the system who are mostly main actors in any application.

3 Modeling and Development Methodology for Agent Design

The developed methodology follows CommonKADS methodology [4]. However CommonKADS is not tied with any modeling tool, knowledge representation or ontologies, we represent knowledge based on the OWL ontology and we model it in the Protégé ontology editor similar as in [4]. Design of the system is based on UML, AUML and MAScommonKADS. When using this methodology, good results can be archived only after several iterations of the process and remodeling after the first developing, use and evaluation of the first system version. Modeling a knowledge model for an application we follow first three CommonKADS models: Organizational or in our case the Environment Model; the Task Model; the Agent or the Actor Model. Modeling the knowledge model we have to extend the generic agent model (Figure 2) with new elements and relations. Results of models is the knowledge model which consists of :

- Ontology developed in Protégé in the OWL format.
- Algorithms for each agent (actor) updating actors' context f_C (2) and resources f_R (3). Often such algorithms are similar or the same for all actors.

The design methodology of a system is based on three UML diagram types in a similar way as known in object oriented programming: Use Case Diagram; Sequence Diagram; Class Diagram.

4 Design and Specification of Agent Software Library

The developed library is based on the JADE agent system [1] and the Jena [13]. It covers functionalities which we identified as missing in current agent architectures such as JADE, the Agent Knowledge Model based on RDF/OWL; Action resource as basics for communication and incorporated in the OWL ontology; Sending ACL based RDF/OWL messages; Receiving ACL based RDF/OWL messages; Incorporation of received information into a model; XMLRPC receiving messages; XMLRPC returning RDF and XML; An inference or RDQL messages handling. The JADE based agent can be developed using this library to support the Jena OWL model as the Agent memory and furthermore it is possible to include XML-RPC based functionalities for presenting knowledge or receiving events from external systems as RDF messages based on a used ontology. Moreover it allows communication among agents based on RDF/OWL.

The developed library is published on the JADE website as the 3rd party software [5] as a way to put together Jena RDF/OWL and JADE Multi-agent system features. The developed library is quite popular and in year 2005 was downloaded 577 times.

5 CONCLUSION

The paper describes how semantic web technologies can be applied in MAS. An agent knowledge model was created enabling the possibility to model the agent environment, agent context and its results. The agent library to support such an agent knowledge model was developed and extends the JADE agent system, which is currently the most popular MAS toolkit. It was proved that such model and the agent architecture are implementable and can be used in different knowledge intensive applications [6], [8] in discrete environments, which was tested in several R&D projects, mainly in the Pellucid IST project. More details with demonstration examples can be found in [8] or on library website [5].

ACKNOWLEDGEMENTS

This work is supported by projects K Wf-Grid IST FP6-511385, SPVV 1025/2004, APVT-51-024604, VEGA No. 2/6103/6.

REFERENCES

- [1] Telecom Italia, *JADE Website*, <http://jade.cselt.it/>, 2004.
- [2] G. Caire, *JADE Tutorial Application-defined Content Languages and Ontology*, <http://jade.cselt.it/>, 2002
- [3] M. Luck, P. McBurney, C. Preist, *Agent Technology:Enabling Next Generation Computing, A Roadmap for Agent Based Computing*, 2003
- [4] G. Schreiber, M. Crubézy, M. Musen, *A Case Study in Using Protege-2000 as a Tool for CommonKADS*, 2001
- [5] M. Laclavik, *AgentOWL: OWL Agent memory model*, 2005, <http://jade.tilab.com/community-3rdpartysw.htm#AgentOWL>
- [6] M. Laclavik et al., *Experience Management Based on Text Notes (EM-BET)*, Innovation and the Knowledge Economy, Volume 2, Part 1; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0,2005
- [7] F. Baader et al., *Description Logic Handbook*, ISBN:0521781760, 2003
- [8] M. Laclavik: *Ontology and Agent based Approach for Knowledge Management*; PhD Thesis; II SAS, field: Applied Informatics, June 2005
- [9] W3C, *OWL*, 2004, <http://www.w3.org/TR/owl-features/>
- [10] M. Laclavik et al., *Methods for Presenting Ontological Knowledge to the User*, Proceedings of Znalosti 2005, pp.61-64. ISBN 80-248-0755-6
- [11] D. Wooldridge, *Introduction to MAS*, ISBN:047149691X, 2002
- [12] Foundation for Intelligent Physical Agent, 2002, <http://www.fipa.org/>
- [13] HP Labs, *Jena Semantic Web Library*, 2004, <http://www.sf.net/>
- [14] J. Kitowski et al., *Model of Experience for Public Organizations with staff Mobility*, KMGov2004, Krems, Austria, May 17-19, 2004

Partial Local FriendQ Multiagent Learning: Application to Team Automobile Coordination Problem¹

Julien Laumonier and Brahim Chaib-draa²

1 Introduction

In real world cooperative multiagent problems, each agent has often a partial view of the environment. If communication has a cost, the multiagent system designer has to find a compromise between increasing the observability and total cost of the multiagent system. To choose a compromise, we propose, to take into account the degree of observability, defined as the agent's vision distance, for a cooperative multiagent system by measuring the performance of the associated learned policy. Obviously, decreasing observability reduces the number of accessible states for agents and therefore decreases the performance of the policy. We restrict our application to team game, a subclass of coordination problems where all agents have the same utility function. We consider problems where agents' designer does not know the model of the world. Thus, we can use learning algorithms which have been proven to converge to Pareto-optimal equilibrium such as Friend Q-learning [4]. One can take an optimal algorithm to find the policy for the observable problem. The following assumptions are defined: (1) Mutually exclusive observations, each agent sees a partial view of the real state but all agents together see the real state. (2) Possible communication between agents but not considered as an explicit part of the decision making. (3) Only negative interactions between agents. One problem which meets these assumptions is the choosing lane decision problem related to Intelligent Transportation Systems which aims to reduce congestion, pollution, stress and increase safety of the traffic.

2 Formal Model and Algorithms

Reinforcement learning allows an agent to learn by interacting with its environment. For a mono agent system, the basic formal model for reinforcement learning is a Markov decision process. Using this model, Q-Learning algorithm calculates the optimal values of the expected reward for the agent in a state s if the action a is executed. On the other hand, game theory studies formally the interactions of rational agents. In a one-stage game, each agent has to choose an action to maximize its own utility which depends on the others' actions. In game theory, the main solution concept is the Nash equilibrium which is the best response for all agents. A solution is Pareto optimal if there does not exist any other solution such that one agent can improve its reward without decreasing the reward of another. The model which combines reinforcement learning and game theory, is

Markov games. This model contains a set of agents Ag , a finite set of states S , a finite set of actions A , a transition function \mathcal{P} , an immediate reward function \mathcal{R} . Among the algorithms which calculate a policy for team Markov games, Friend Q-Learning algorithm, introduced by Littman [4], allows to build a policy which is a Nash Pareto optimal equilibrium in team games. More specifically, this algorithm, based on Q-Learning, uses the following function for updating the Q-values: $Q(s, \vec{a}) = (1 - \alpha)Q(s, \vec{a}) + \alpha[r + \gamma \max_{\vec{a} \in \vec{A}} Q(s', \vec{a})]$ with \vec{a} , the joint action for all agents.

3 Problem Description

The vehicle coordination problem presented here is adapted from Moriarty and Langley [5]. More precisely, three vehicles have to coordinate to maintain velocity and to avoid collisions. Each vehicle is represented by a position and a velocity and can change lane to the left, to the right or stay on the same lane. The objective for a learning algorithm is to find the best policy for each agent in order to maximize the common reward which is the average velocity at each turn and to avoid collision. The dynamic, the state and the actions are sampled in the simplest way. For this example, we simulate the road as a ring meaning that a vehicle is placed on the left side when it quits through the right side. Collisions occur when two agents are in the same case. At each step, a vehicle can choose three actions: stay on the same lane, change to the right lane and change to the left lane. We assume, in this problem, that each agent is able to see only his local state and other's states with communication.

4 Partial Observability

To measure the effect of partial observability on the performance we define the partial state centered on one agent by introducing a distance of observability d . The initial problem becomes a d -partial problem. The distance d can be viewed as an influence area for the agent. Increasing this distance increases the degree of observability. We define d_{total} as the maximal possible distance of observability for a given problem. In d -partial problem, the new state is defined as the observation of the center agent for a range d . More precisely, an agent j is in the partial state of a central agent i if its distance is lower or equal than d from i . The function $f_d^i(s)$ uses the parameter d to calculate the new local state. The partial local view can reduce the set of state and/or the set of joint actions. The new size of the set of state is set from $O((X \times Y \times |V|)^N)$ with X the number of lane, Y the length of the road, V the set of possible velocity and N the number of agents to $O(((2d + 1)^2 \times V)^N)$. The number of states is divided by around $(Y/(2d + 1))^N$. The Partial Joint Action (PJA) algorithm takes into account only the actions of agents that are in the partial local view as specified by d . This reduces dramatically

¹ This research is funded by the AUTO21 Network of Centers of Excellence, an automotive research and development program focusing on issues relating to the automobile in the 21st century. AUTO21 is a member of the Networks of Centers of Excellence of Canada program.

² DAMAS Laboratory, Department of Computer Science and Software Engineering, Laval University, Canada {jlaumoni;chaib}@damas.ift.ulaval.ca

the number of joint actions which has to be tested during the learning. This partial local observability allow us to consider a variable number of agents in the multiagent system. Formally, we define a function g^i which transforms the joint action \vec{a} into a partial joint action $g_d^i(\vec{a}, s)$. This partial joint action contains all actions of agent which are in the distance d of agent i . The PJA Q-value update function is :

$$\begin{aligned} Q(f_d^i(s), g_d^i(\vec{a}, s)) &= (1 - \alpha)Q(f_d^i(s), g_d^i(\vec{a}, s)) \\ &\quad + \alpha[r + \gamma \max_{\vec{a}_d \in G_d^i(\vec{A}, S)} Q(f_d^i(s'), \vec{a}_d)] \end{aligned}$$

5 Results

We compare empirically the performance of the totally observable problem (FriendQ) and the performance of approximated policy (PJA) on a small problem defined by size $X = 3, Y = 7$ with 3 agents. Figure 1 shows that for $d = 0 \dots 2$, PJA converges to a local maximum, which increases with d . In these cases, the approximated values are respectively about 76%, 86% and 97% from the optimal value given by Friend Q-Learning. When $d = 3$, that is, when the local view is equivalent to the totally observable view, the average sum rewards converges to the total sum rewards of Friend Q-learning.

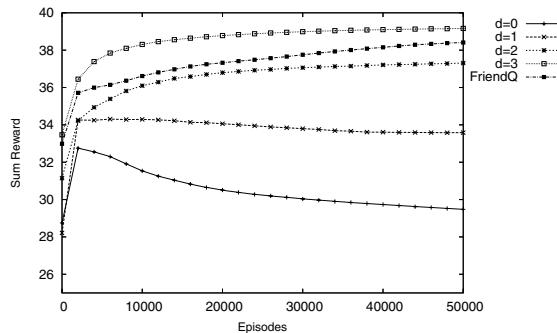


Figure 1. Rewards for Partial Joint Action Q-learning

The generalization of these results can be done applying PJA on larger problems. Calculating the near optimal policy, using an approximated optimal distance d_{app} , can be intractable if we need to compare with the results of Friend Q-Learning. We calculate the ratio $DS = XY/N$ which represents the degree of space for each agent. As we study a problem where the team of agent has to handle only negative interactions, the higher the ratio, the more space agents have. We compare the performance of PJA algorithm for different ratios. To discover a relation between the ratio DS and the value of d_{app} , we compare the link between DS and $\frac{d_{app}}{d_{total}}$. We can see that the degree of observability decreases with the degree of space for each agent and that the needed observability decreases and tends to 0 when DS increases. With this relation between both parameter observability and degree of space, we can evaluate, for other problems how would be the d_{app} value.

6 Related Work

Concerning the space search reduction, Sparse Cooperative Q-Learning [3] allows agents to coordinate their actions only on predefined sets of states. The joint actions set reduction has been studied

by Fulda and Ventura who propose Dynamic Joint Action Perception [2]. Introducing communication into decision has been studied by Xuan, Lesser, and Zilberstein [8] who proposed a formal extension to MDP with communication and by Pynadath and Tambe [7] who proposed an extension to distributed POMDP with communication called COM-MTDP, which take into account the cost of communication during the decision process. The locality of interactions in an MDP has been theoretically developed by Dolgov and Durfee [1]. Regarding the reinforcement learning in a vehicle coordination problem, Ünsal, Kachroo and Bay [9] have used multiple stochastic learning automata to control longitudinal and lateral path of one vehicle and Pendrith [6] presented a distributed variant of Q-Learning applied to lane change advisory system.

7 Conclusion

We proposed an approach to evaluate a good approximation of a multiagent decision process, introducing a degree of observability for each agents. Without taking into account explicit communication to obtain a degree of observability, we proposed Friend Q-learning algorithms extension which uses only observable state and observable actions from the other agents. We show that only partial view is needed to obtain a good policy approximation for some team problems, especially the changing lane problem between vehicles. We show a relation between a good degree of observability and the space allowed for each agent. However, this relation is only empirical and our approach is only restricted to negative interaction management problems. It is possible that in other problems, this relation could be different. For future work, we plan to evaluate more theoretically the relation between the degree of observability and the performance of the learned policy. Also, it will be very interesting to study the effect of partial local view to non cooperative and non negative interactions cases.

REFERENCES

- [1] Dmitri Dolgov and Edmund H. Durfee, ‘Graphical models in local, asymmetric multi-agent Markov decision processes’, in *Proceedings of AAMAS 2004*, (2004).
- [2] Nancy Fulda and Dan Ventura, ‘Dynamic Joint Action Perception for Q-Learning Agents’, in *2003 International Conference on Machine Learning and Applications*, (2003).
- [3] Jelle R. Kok and Nikos Vlassis, ‘Sparse Cooperative Q-learning’, in *Proc. of the 21st ICML*, pp. 481–488, Banff, Canada, (July 2004). ACM.
- [4] Michael Littman, ‘Friend-or-Foe Q-learning in General-Sum Games’, in *Eighteenth ICML*, ed., Morgan Kaufmann, pp. 322–328, (2001).
- [5] David E. Moriarty and Pat Langley, ‘Distributed learning of lane-selection strategies for traffic management’, Technical report, Palo Alto, CA., (1998). 98-2.
- [6] Mark D. Pendrith, ‘Distributed reinforcement learning for a traffic engineering application’, in *the fourth International Conference on Autonomous Agents*, pp. 404 – 411, (2000).
- [7] David V. Pynadath and Milind Tambe, ‘The communicative multiagent team decision problem: Analyzing teamwork theories and models’, *JAIR*, **16**, 389–423, (2002).
- [8] Ping Xuan, Victor Lesser, and Shlomo Zilberstein, ‘Communication decisions in multi-agent cooperation: model and experiments’, in *the Fifth International Conference on Autonomous Agents*, eds., Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, pp. 616–623, Montreal, Canada, (2001). ACM Press.
- [9] Cem Ünsal, Pushkin Kachroo, and John S. Bay, ‘Simulation study of multiple intelligent vehicle control using stochastic learning automata’, *IEEE Transactions on Systems, Man and Cybernetics - Part A : Systems and Humans*, **29**(1), 120–128, (1999).

Mutual Enrichment for Agents Through Nested Belief Change: A Semantic Approach

Laurent Perrussel and Jean-Marc Thévenin¹ and Thomas Meyer²

1 Introduction

This paper focuses on the dynamics of nested beliefs in the context of agent interactions. Nested beliefs represent what agents believe about the beliefs of other agents. We consider the *tell* KQML performative [1] which allows agents to send their own beliefs to others. Whenever agents accept a new belief, or refuse to change their own beliefs after receiving a message, both receiver and sender enrich their nested beliefs by refining their beliefs about (i) the other agent's beliefs and (ii) the preferences of the other agent. The main objective of nested beliefs is to improve cooperation between agents. We propose a logical formalisation of the acquisition process of nested beliefs and preferences. This acquisition process is the first step towards the elaboration of sophisticated interaction protocols.

2 The Logical Framework

To represent an agent's beliefs we use signed statements. A *signed statement* is a pair $\langle \text{statement}, \text{origin of the statement} \rangle$ (usually the sender of the statement). Let \mathcal{L}_0 be a propositional language and A a set of agent identities. We define a signed statement as a pair $\langle \phi_0, a \rangle$ where ϕ_0 is a \mathcal{L}_0 -formula and $a \in A$ is the origin of ϕ_0 . Let S be the set of all sets of signed statements: $S = 2^{\mathcal{L}_0 \times A}$. The *belief state* of an agent is a pair $\langle \text{set of signed statements}, \text{set of sets of signed statements} \rangle$. The first set describes the *basic beliefs* of the agent: what it currently believes. The second set describes the *nested beliefs* of the agent: what it believes about the basic beliefs of others.

Definition 1 (Belief state) A belief state BS_a^n of agent a is a pair $\langle CB_a^n, NB_a^n \rangle$ s.t. (i) $CB_a^n \in S$ represents the basic beliefs of agent a at time n and (ii) $\forall CB_{a,b}^n \in NB_a^n, CB_{a,b}^n \in S$ represents the nested beliefs of agent a about agent b at n . Let \mathcal{B} be the set of all possible belief states.

Agents revise their basic beliefs and nested beliefs each time they receive a *tell* performative. Let S be a set of signed beliefs and $*$ be a revision operator [2]; $S_{\langle \phi_0, a \rangle}^*$ denotes the revision of S by $\langle \phi_0, a \rangle$.

Preferences may be defined taking various matters into account [5]. We simply assume that agents have preferences over the set of

agents A which describe the reliability of the sources of information, i.e. the level of trust an agent has about the other agents with which it interacts. We suppose that agents are equally reliable when they can't be distinguished, which entails a total preorder. Let \preceq_a^n be a total preorder over A representing agent a 's preferences at time n . $b \preceq_a^n c$ stands for agent c is at least as preferred as b for agent a at time n . As is the case for beliefs, agents can handle nested preferences. Nested preferences represent what agents believe about the preferences of other agents. $c \preceq_{a,b}^n d$ means: agent a believes that for agent b agent d is at least as preferred as c at n .

Definition 2 (Preference state) A preference state PS_a^n of agent a is a pair $\langle \preceq_a^n, NP_a^n \rangle$ s.t. (i) \preceq_a^n is a total preorder representing basic preferences of agent a at time n and (ii) every $\preceq_{a,b}^n \in NP_a^n$ is a total preorder representing agent b 's preferences according to a at time n . \mathcal{P} is the set of all possible preference states.

Let $\langle BS_a^n, PS_a^n \rangle = \langle \langle CB_a^n, NB_a^n \rangle, \langle \preceq_a^n, NP_a^n \rangle \rangle$ be a tuple which represents the whole state of agent a at n , i.e. all its beliefs and preferences.

Let S be any set of signed statements (basic or nested beliefs) and \preceq the corresponding preference relation (basic or nested). The logical closure of S w.r.t. \preceq is obtained as follows. By $\min(S, \preceq)$ we denote the set of the least preferred agent identities w.r.t. \preceq among agent identities signing beliefs of S . We suppose that statements entailed by S are signed with the least preferred agent identities of the minimal subsets of S entailing them: $Cn(S, \preceq) = \{ \langle \psi_0, a \rangle | \exists S' \subseteq S \text{ s.t. } \bigwedge_{\langle \phi_0, b \rangle \in S'} \phi_0 \models_{\mathcal{L}_0} \psi_0 \text{ and } \#S'' \subset S' \text{ s.t. } \bigwedge_{\langle \phi_0, b \rangle \in S''} \phi_0 \models_{\mathcal{L}_0} \psi_0 \text{ and } a \in \min(S', \preceq) \} \cup \{ \langle \psi_0, a \rangle | \models_{\mathcal{L}_0} \psi_0 \text{ and } a \in A \}$.

Now, we present the *action performatives* which lead to the dynamics of belief and preference states change. When an agent issues a *tell* performative to inform a receiver agent about its basic beliefs the receiver uses a prioritised belief revision operator $*$ to change its nested beliefs about the sender. As an acknowledgment of the *tell* performative, the receiver informs the sender with an *accept* (respectively *deny*) performative if the incoming statement has been incorporated in its basic beliefs. The sender in turn applies prioritised revision to its nested beliefs about the receiver. More formally:

- $\text{Tell}(s, r, \phi_0, a)$ stands for: agent s informs r that it believes ϕ_0 signed by a according to the standard KQML semantics [1]. When receiving a *Tell* performative, agent r revises its belief state by $\langle \phi_0, a \rangle$ in a non-prioritised way [3] according to its preferences.
- $\text{Accept}(r, s, p)$ stands for: agent r informs s that it accepts the performative p . If $p = \text{Tell}(s, r, \phi_0, a)$ then $\text{Accept}(r, s, p)$ means that agent r has revised its basic beliefs by $\langle \phi_0, a \rangle$ and thus believes ϕ_0 .
- $\text{Deny}(r, s, p)$ stands for: agent r informs s that it refuses to process

¹ IRIT-Université Toulouse 1, Manufacture des Tabacs, 21 allée de Brienne, F-31042, Toulouse Cedex - France, email: laurent.perrussel@univ-tlse1.fr, email: jean-marc.thevenin@univ-tlse1.fr

² National ICT Australia and University of New South Wales, Sydney, Australia, email: thomas.meyer@nicta.com.au. National ICT Australia is funded by the Australia Government's Department of Communications, Information and Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program. It is supported by its members the Australian National University, University of NSW, ACT Government, NSW Government and affiliate partner University of Sydney.

the performative p . If performative $p = \text{Tell}(s, r, \phi_0, a)$ it means that agent r has not revised its basic beliefs by $\langle\phi_0, a\rangle$.

The dynamics of the system is given by the execution of performatives. A sequence of actions σ is a function which associates integers representing state labels with performatives.

3 The Dynamics of Mutual Enrichment

Let us first express the honesty postulate (Hon) as follows. This postulate, which is recommended in a cooperative context, states that if a Tell performative occurs, then at the same time the sender believes the corresponding signed statement. This actually enforces the standard KQML semantics of Tell [1].

(Hon) For any n if $\sigma(n) = \text{Tell}(s, r, \phi_0, a)$ then $\langle\phi_0, a\rangle \in \text{Cn}(CB_s^n, \preceq_s^n)$.

Basically, after a performative $\text{Tell}(s, r, \phi_0, a)$, the belief state and preference state of agents do not change, except for the receiver. The receiver applies a non-prioritised revision of its basic beliefs using its basic preferences. It also applies a prioritised revision of its nested beliefs about the sender since it believes ϕ_0 signed by a according to the honesty postulate. It can also refine its nested preferences about the sender if it has removed a nested belief $\neg\phi_0$ signed by an agent b during the prioritised revision. Indeed the sender has preferred a to b . Finally the receiver acknowledges the Tell performative with an Accept or Deny performative, as formalized below, depending on the outcome of the non-prioritised revision of its basic beliefs so that the sender can in turn change its nested beliefs and nested preferences about the receiver. The non-prioritised revision is based on basic preferences in the following way: if the receiver believes $\neg\phi_0$ and the signatures of $\neg\phi_0$ are at least as preferred as a (the signature of ϕ_0 in the Tell), the receiver denies the Tell performative.

(DT) $\sigma(n+1) = \text{Deny}(r, s, \text{Tell}(s, r, \phi_0, a))$ iff $\sigma(n) = \text{Tell}(s, r, \phi_0, a) \& \exists \langle\neg\phi_0, b\rangle \in \text{Cn}(CB_r^n, \preceq_r)$ s.t. $a \preceq_r b$

Otherwise there is no conflict or a is strictly more preferred than all the signatures of $\neg\phi_0$, and r thus accepts the Tell performative.

(AT) $\sigma(n+1) = \text{Accept}(r, s, \text{Tell}(s, r, \phi_0, a))$ iff $\sigma(n) = \text{Tell}(s, r, \phi_0, a) \& \forall \langle\neg\phi_0, b\rangle \in \text{Cn}(CB_r^n, \preceq_r), b \prec_r a$

Due to space restrictions we focus on the formulas describing how nested beliefs and nested preferences of the sender change (see [6] for the complete formalisation of the dynamics).

According to (AT), if r accepts the message of s , agent s believes that r believes ϕ_0 and thus does not believe $\neg\phi_0$, which results in s revising its nested beliefs about r with $\langle\phi_0, a\rangle$ (a prioritised revision).

(AdNB3-1) If $\sigma(n) = \text{Accept}(r, s, \text{Tell}(s, r, \phi_0, a))$ then $CB_{s,r}^{n+1} = (CB_{s,r}^n)_{\langle\phi_0, a\rangle}^*$.

According to (DT), if r refuses the Tell performative then s concludes that r believes $\neg\phi_0$ and r believes that the signature of $\neg\phi_0$ has to be more trusted than a . So s revises its nested beliefs about r with the signed statement $\langle\neg\phi_0, b\rangle$ (again, a prioritised revision) so that signature b of $\neg\phi_0$ is preferred to a w.r.t. the nested preferences of s about r :

(AdNB3-2) If $\sigma(n) = \text{Deny}(r, s, \text{Tell}(s, r, \phi_0, a))$ then $CB_{s,r}^{n+1} = (CB_{s,r}^n)_{\langle\neg\phi_0, b\rangle}^*$ s.t. $a \preceq_{s,r} b$

This bring us to the sender's nested preferences. Firstly, the sender's nested preferences about agents other than r do not change. Next, agent s draws conclusions about a depending on whether r accepts or denies the performative $\text{Tell}(s, r, \phi_0, a)$. Whenever agent r accepts, agent s refines its nested preferences only if s currently believes that r believes $\neg\phi_0$ so that a be strictly more preferred than the signatures of $\neg\phi_0$. If r denies the message, the nested preferences of s about the signatures of $\neg\phi_0$ change. According to condition (DT) they have to be as preferred as a . Finally, we need to take care of nested preferences, about r , of s that do not change. In order to formalise these requirements, we give an explicit procedure to change the nested preferences. The Accept performative helps the sender to refine its nested preferences since it allows to remove some preferences; i.e. it helps agent s to go toward a stricter order. Removing preferences means that agent s already believes that r believes $\neg\phi_0$ and thus a is strictly preferred to all the signatures of $\neg\phi_0$.

(Ad-KeNP2) Let $\langle CB_{s,r}^n, \preceq_{s,r}^n \rangle$ be the nested beliefs and preferences of s about r . Let $\sigma(n) = \text{Tell}(s, r, \phi_0, a)$ and $\sigma(n+1) = \text{Accept}(r, s, \text{Tell}(s, r, \phi_0, a))$. All nested preferences at n are propagated at time $n+1$ as follows: $\preceq_{s,r}^{n+1} = \preceq_{s,r}^n - \{a \preceq_{s,r} b | \langle\neg\phi_0, b\rangle \in CB_{s,r}^n\} \cup \{b \preceq_{s,r} a | \langle\neg\phi_0, b\rangle \in CB_{s,r}^n\}$.

If r does not accept the incoming message, agent s also changes its nested beliefs about r . The Deny performative does not help agent s to refine its nested preferences since we only add nested preferences.

(AdNP3) Let $\langle CB_{s,r}^n, \preceq_{s,r}^n \rangle$ be the nested beliefs and preferences of s about r . Let $\sigma(n) = \text{Tell}(s, r, \phi_0, a)$ and $\sigma(n+1) = \text{Deny}(r, s, \text{Tell}(s, r, \phi_0, a))$. All nested preferences at n are propagated at time $n+1$ as follows: $\preceq_{s,r}^{n+1} = \preceq_{s,r}^n \cup \{a \preceq_{s,r} b | \langle\neg\phi_0, b\rangle \in CB_{s,r}^n\}$.

It is easily shown that the conditions preserves the ordering for nested preference as a total preorder.

4 Conclusion

In this paper we have given a sketch of a formalisation for handling the dynamics of nested beliefs and preferences in the context of agent interactions where agents are cooperative (see [6] for the detailed formalisation). We have shown how agents acquire nested beliefs and preferences. For this we have presented a logical framework to describe nested beliefs, preferences, and performatives. This framework is useful for specifying properly the expected behaviour of agents handling the Tell , Accept and Deny performatives. We have started to build a logical language based on dynamic epistemic logic [4] in order to reason about dialogues. Our aim is to define a semantics based on the semantics proposed in this paper.

REFERENCES

- [1] T. Finin, Y. Labrou, and J. Mayfield, 'KQML as an agent communication language', in *Software Agents*, ed., J. Bradshaw, MIT Press, (1997).
- [2] P. Gärdenfors, *Knowledge in flux: Modeling the Dynamics of Epistemic States*, MIT Press, 1988.
- [3] S. O. Hansson, 'A survey of non-prioritized belief revision', *Erkenntnis*, **50**, 413–427, (1999).
- [4] J.-J. C. Meyer and W. van der Hoek, *Epistemic Logic for AI and Computer Science*, Cambridge University Press, 1995.
- [5] L. Perrussel and J.M. Thévenin, 'A logical approach for describing (dis)belief change and message processing', in *Proceedings of AAMAS'04*, pp. 614–621. IEEE C.S., (2004).
- [6] L. Perrussel, J.M. Thévenin, and T. Meyer, 'Mutual enrichment for agents through nested belief change: A semantic approach', in *Proc. of NMR'06*, (2006).

Multi-Agent Least-Squares Policy Iteration

Victor Palmer¹

Abstract. Least-Squares Policy Iteration [3] is an approximate reinforcement learning technique capable of training policies over large, continuous state spaces. Unfortunately, the computational requirements of LSPI scale poorly with the number of system agents. Work has been done to address this problem, such as the Coordinated Reinforcement Learning (CRL) approach of Guestrin, et al [1], but this requires that one have prior information about the learning system such as knowing interagent dependencies and the form of the Q -function. We demonstrate a hybrid gradient-ascent/LSPI approach which is capable of using LSPI to efficiently train multi-agent policies. Our approach has computational requirements which scale as $O(N)$, where N is the number of system agents, and does not have the prior knowledge requirements of CRL. Finally, we demonstrate our algorithm on a standard multi-agent network control problem [1].

1 Introduction

Least-Squares Policy Iteration [3] is an approximate reinforcement learning algorithm capable of efficiently training policies over large, continuous state spaces. Unfortunately, like almost all reinforcement learning algorithms, LSPI has computational requirements which scale exponentially with the number of system agents, making it difficult to use the algorithm in a multi-agent setting. This exponential scaling is due to the fact that the action-space for a multi-agent system scales exponentially with the number of system agents. Work has been done to address this problem, most notably the Coordinated Reinforcement Learning approach of Guestrin, et. al. [1], although this suite of techniques requires that we know prior information about interagent dependencies and that we make assumptions about the form of the joint state-action valuation (discounted reward) function Q . Here we introduce a gradient-based 'wrapper' to LSPI which allows the algorithm to be performed efficiently on multi-agent domains.

2 Reinforcement Learning / LSPI Review

We work in the standard reinforcement learning context [4], where a system of agents $n_i \in \mathcal{N}$ interacts with an environment modeled by an MDP. The environment can take on states $s_t \in \mathcal{S}$, and at each time step, each agent can perform actions $a_t \in \mathcal{A}$. The joint action of our agent system will be denoted as the vector $\mathbf{a}_t \in \mathcal{A}^N$, where each component represents the action of a single agent. Let our policy be denoted $\pi(\mathbf{a}; s)$, which is the probability of performing actions \mathbf{a} when the environment is in state s . We will use the standard notion of discounted reward $Q^\pi(s, a)$, and also the *average reward* $\eta(\pi) = \sum_{s,a} \rho^\pi(s) \pi(a; s) R(s, a)$, where $\rho^\pi(s)$ is the stationary probability distribution of the policy, as in [2].

¹ Texas A&M University, College Station, Texas, email: vpalmer@cs.tamu.edu

In many modern control domains, it is not possible to explicitly represent the discounted reward function Q . Assuming that we approximate Q as: $Q^\pi(s, a) \approx \hat{Q}^\pi(s, a; \vec{\omega}) = \sum_{i=1}^k \phi_i(s, a) \omega_i$, [3] shows us how we can find the $\vec{\omega}$ such that the error $\sum_{s,a} |Q^\pi(s, a) - \hat{Q}^\pi(s, a; \vec{\omega})|$ is minimized. Specifically, [3] shows how to solve for the error-minimizing $\vec{\omega} = \Omega(\pi)$ by creating a policy-dependent matrix $\mathbf{A}(\pi)$, and a vector \mathbf{b} and solving the system $\mathbf{A}(\pi)\vec{\omega} = \mathbf{b}$ for $\vec{\omega}$. Solving this system performs *policy valuation*, after which an immediately improved policy can be constructed by: $\pi'(s) = \arg \max_a \hat{Q}(s, a; \vec{\omega})$. This new policy can be used to construct a new matrix $\mathbf{A}(\pi')$, and the process can be repeated to perform approximate policy iteration [3]. Unfortunately, the matrix \mathbf{A} ends up being proportional to the size of the joint action space, which grows exponentially with the number of agents in the system. As such, it is difficult to perform LSPI in large multi-agent domains.

3 Policy Gradient Ascent

An alternative to value-based reinforcement learning algorithms such as LSPI is the class of gradient-based algorithms such as REINFORCE [5] or the Conjugate Policy Gradient techniques of Kakade, et. al. [2]. Such algorithms directly search the space of policies by performing updates in the direction of some objective function (usually the average reward $\eta(\pi)$) gradient. Recently an interesting connection was drawn between the gradient of the average reward and approximate policy iteration techniques such as LSPI. Specifically, it was shown that $\nabla \eta(\pi(\omega)) = \Omega(\pi(\omega))$ [2].

Imagine that our joint policy conforms to the following ansatz:

$$\pi(\mathbf{a}; s) = \pi_{n_1}(\mathbf{a}_1; s) \pi_{n_2}(\mathbf{a}_2; s) \dots \pi_{n_N}(\mathbf{a}_N; s) \quad (1)$$

This models the case where each agent's policy is executed independently. Note that this is automatically true for deterministic policies. If we want to perform gradient ascent on the average reward $\eta(\pi)$ of this joint policy, we need to compute:

$$\nabla \eta(\pi) = \nabla \eta(\pi_{n_1}, \pi_{n_2}, \dots, \pi_{n_N}) \quad (2)$$

In an LSPI context, each of these single agent policies can be parameterized by a parameter vector $\vec{\omega}_{n_i}$ as $\pi_{n_i} = \pi_{n_i}(\vec{\omega}_{n_i}) = \arg \max_{a'} \hat{Q}(s, a'; \vec{\omega}_{n_i})$ so that we have (since the policy parameters are just the $\vec{\omega}_{n_i}$):

$$\nabla \eta(\pi) = \sum_{i=1}^N \nabla_{\vec{\omega}_{n_i}} \eta(\pi) \quad (3)$$

$$\nabla \eta(\pi) = \sum_{i=1}^N \nabla_{\vec{\omega}_{n_i}} \eta\left(\pi_{n_1}(\vec{\omega}_{n_1}), \dots, \pi_{n_N}(\vec{\omega}_{n_N})\right) \quad (4)$$

but by the chain rule:

$$\nabla \eta(\pi) = \sum_{i=1}^N \nabla_{\omega_{n_i}} \eta(\pi_{n_i}(\vec{\omega}_{n_i})) \quad (5)$$

where, for the computation of each gradient $\nabla_{\omega_{n_i}} \eta(\pi_{n_i})$, the other agents follow an unaltered policy:

$$\forall j \neq i, \quad \pi_{n_j} = \pi_{n_j}(\vec{\omega}_{n_j}) \quad (6)$$

Because each of the $\nabla \eta$ terms are single-agent policy $\eta(\pi)$ expressions, the reader should note that at this point, our original, multi-agent gradient computation task has been decomposed into N , single-agent gradient computation tasks. Further, since single-agent LSPI can be used to compute each of the single-agent gradients, we do not run into the problem of the exponential computational requirements of performing LSPI over multiple agents. Since each iteration of policy iteration scales as $O(k^3)$ [3], where k is the number of basis functions used for approximation, this hybrid gradient-ascent/LSPI approach should scale as $O(Nk^3)$ per iteration ... as opposed to $O(C^N)$ as if LSPI were used directly to train all N agents.

4 Experiments

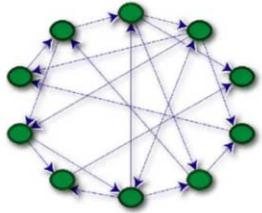


Figure 1. This figure shows a typical, randomly generated network used in our simulations (an $N = 10$ network is shown)

We tested this gradient-ascent/LSPI approach on the SysAdmin domain, which is a multi-agent network control problem [1]. In this domain, a network of computers are managed by a system of N agents (one agent per computer). Agents get reward when processes complete on their computers, but healthy computers can become faulty and then fail, and faulty computers execute tasks more slowly, and failed computers execute no tasks at all. Further, connected faulty and failed machines may 'infect' their connected neighbors, making them fail as well. An agent may 'reboot' its machine and to return it to 'healthy' status, but at the expense of losing all running processes. The goal is for all agents to coordinate their actions to maximize total reward. A nice property of this problem is that it has an easily calculable 'utopic upper bound' on total reward (basically assuming that no computer ever infects its neighbors, and that each machine is rebooted appropriately) that good policies can come close to achieving [1]. We generated random N -computer networks and created random links between the nodes such that each node had exactly two incoming connections. This was done to emulate the 'ring-of-rings' network topology used in the CRL paper [1].

We trained joint policies using our gradient ascent technique. All experiment parameters were as in [1]. Following the approach of [3], we used $k = 10N$ basis functions to approximate the single-agent $Q^\pi(s, a)$ function. We found that as predicted, training time scaled polynomially (specifically, as $O(N^4)$ here since $k \propto N$), and that the policies were of comparable quality to those generated by CRL

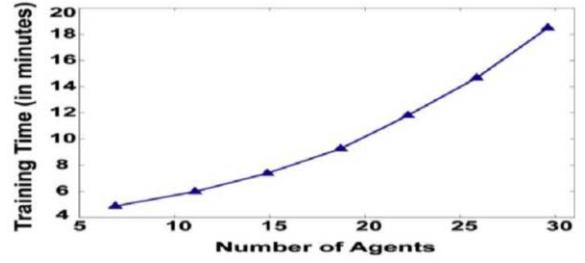


Figure 2. The total training time required to train our networks of various sizes. We predicted this to scale as $O(N^4)$, and in fact, dividing each point by N^4 results in a monotonically decreasing function (not shown).

[1]. Note that unlike in CRL, here we did not have to integrate knowledge of interagent dependencies into the learning algorithm.

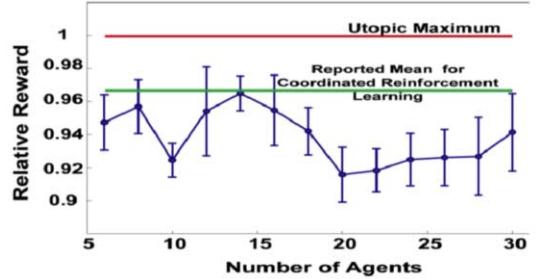


Figure 3. Shows the (relative) average reward of the final policies obtained by our gradient ascent technique. All rewards are relative to the utopic maximum as described in the Guestrin work. We plot the mean reward (over all agent counts) reported in [1] for CRL on the ring-of-rings network.

5 Conclusions

We have shown how a hybrid gradient-ascent approach can be used to train a multi-agent system with LSPI. The approach scales computationally as $O(N)$ where N is the number of agents, and can be used without prior knowledge of interagent dependencies. We tested our approach on the SysAdmin multi-agent network control problem [1] and found that our policies were competitive with those generated by Coordinated Reinforcement Learning [1].

ACKNOWLEDGEMENTS

This work was supported by a Fannie and John Hertz Foundation fellowship.

REFERENCES

- [1] D. Koller D. Guestrin and R. Parr, 'Multiagent planning with factored mdps', in *NIPS-14*, (2001).
- [2] S. Kakade, 'A natural policy gradient', in *Advances in Neural Information Processing Systems (NIPS14)*, (2002).
- [3] M. G. Lagoudakis and R. Parr, 'Least-squares policy iteration', *Journal of Machine Learning Research*, **4**, 1107–1149, (2003).
- [4] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
- [5] R. Williams, 'Simple statistical gradient-following algorithms for connectionist reinforcement learning', *Machine Learning*, **8**, 229–256, (1992).

4. Knowledge Representation and Reasoning

This page intentionally left blank

Finding Instances of Deduction and Abduction in Clinical Experimental Transcripts

Maria Amalfi¹, Katia Lo Presti², Alessandro Provetti³ and Franco Salvetti⁴

Abstract. This article describes the design and implementation of a prototype that analyzes and classifies transcripts of interviews collected during an experiment that involved lateral-brain damage patients. The patients' utterances are classified as instances of categorization, prediction and explanation (abduction) based on surface linguistic cues. The agreement between our automatic classifier and human annotators is measured. The agreement is statistically significant, thus showing that the classification can be performed in an automatic fashion.

1 Introduction

This article describes a software that we have designed and implemented in the context of our work on measuring utterances that are evidence of inferential ability, viz., prediction and explanation, trying to match a human annotator. The starting point was an experiment that involved patients with lateral-hemisphere brain damage (along with a control group) that was carried out at the Boston VA hospital and for which the results were available in writing. The experiment consisted in showing to the subject the picture in and asking them "*I have a picture here. As you see, there's a lot going on. Look it over and tell me about it.*"

In this article we are not concerned with the cognitive interpretation of the data, nor with the validation of a particular hypothesis relating lateral brain damage to particular types of reasoning impairment. Rather, we would like to validate our approach to automated text classification by showing, via statistical analysis, that the results are comparable with those of human annotators. Our architecture is designed to avoid commitment to any particular model of rationality but could serve as a tool for validating Cognitive Science theories.

Indeed, one could say that the relatively simple software architecture described here is effective for textual analysis only when simple sentences having a limited lexicon are considered. However, the advantage of using an automated tool will be felt when similar experiments will be administered to large populations and human annotation will become uneven or even impossible. Although our software, which has SWI Prolog as its core and is described next, is not suitable for large-scale activities *as is*, standard computational complexity analysis yields that our approach can indeed scale up to several hundred transcripts.

Let us now describe in detail the architecture and the data representation adopted in this project⁵. The system consists of two main

¹ Grad. Student at University of Messina, Italy. maria.amalfi@gmail.com

² Grad. Student at University of Messina, Italy. katia.lopresti@gmail.com

³ Dept. of Physics University of Messina, Italy. E-mail: ale@unime.it

⁴ Umbria Inc. Boulder CO U.S.A. E-mail:franco.salvetti@umbrialistens.com

⁵ The software (source and binary codes), the results and documentation is available from our group page: <http://mag.dsi.unimi.it/>

components that have been designed and implemented for the experiment described above.

1. program *patients-aggregator* takes as input data:

- the patients' transcript;
- local vocabulary⁶ and
- the general-purpose deduction rules for the Prolog inferential engine

and produces:

- rules data about categorizations, explanations and predictions;
- patients' data

Patients-aggregator scans the transcripts and creates a suitable Prolog representation of the phrases. Then, it also produces the schematic rules that the subsequent Prolog interpretation will use to discover the instances of prediction and explanation in the transcripts.

2. program *inference-finder* is written in Prolog; it takes as input the data generated by patients-aggregator and it produces:

- statistics;
- patient predictions and explanations and
- firing rules.

2 Results and comparisons

All the interviews considered in this work were annotated by two independent panels of human annotators, here called B and M. Each panel was made of two graduate students of Computer Science, who received similar instructions and very precise instructions on how to annotate interviews.

2.1 Validation of the results

The overall No. of annotations obtained during our experiment is illustrated in Table 1.

In the following we described a statistical analysis of the results that supports a more sophisticated understanding of the results.

The Kappa index [1], introduced by Cohen [2], has been proposed as a measure of the specific agreement for category among two observers. Kappa measures the accord among the answers of two observers (or the same observer in different moments), that appraises couples of objects or diagnostic categories.

⁶ The number of words of interest is finite and rather limited, i.e., 149 words as categorizations. So, it has been possible to represent all tokens of interest by means of Prolog facts.

Instances Found		
-	expl.	pred.
B panel	99	71
M panel	87	35
Program	71	32

Table 1. Overall no. of instances found

Predictions			
-	B	M	Program
B panel	1	0,244	0,260
M panel	-	1	0,447
Program	-	-	1

Table 3. The K degree of agreement on predictions

This index captures and corrects the so-called *accidentals agreements*. An agreement is called accidental when two observers reach the same conclusion even though they did so by employing completely different sets of criteria to distinguish between the presence/absence of relevant conditions. In such cases the raw agreement index would not reflect a real agreement. The idea underlying the Kappa index is that the actual accord between two observers is as the difference between the raw agreement and the agreement we would have under the hypothesis that between the two there is no accord and thus their answers may coincide only by chance.

The value of K is given by the ratio between the excess agreement ($P_o - P_e$) and the maximum obtainable agreement ($1 - P_e$) :

$$K = \frac{P_o - P_e}{1 - P_e} \quad (1)$$

where:

P_o it is the proportion of frequencies observed of accords among the two evaluation, and

P_e it is the proportion of accords obtained under the condition that the void hypothesis is true.

If there is complete agreement, then K will be equal to 1. If the observed agreement is greater or equal than the agreement attended only by chance obtained then the K index will result near zero or even slightly negative. Values of K above 0.75 suggest that there is an excellent agreement; values below 0.40 represent a weak agreement, whereas values between 0.40 and 0.75 can represent a good agreement. To sum it up, k is the right type of index to assess the quality of our program vis-à-vis human analysis of some experimental results. The degree of agreement among the human panels and the program is as follows:

Explanations			
-	B	M	Program
B panel	1	0,331	0,335
M panel	-	1	0,433
Program	-	-	1

Table 2. The K degree of agreement on explanations

Between the M panel and the system there is a good enough agreement (values between 0.75 and 0.40); while group B has weak agreement both with group M and with the system (values below 0.40).

2.2 Interpretation of the results

The statistics described above show a good agreement between the program scores and those given by the M panel. Vice versa, the B

panel results have a low agreement index K with both the program and the M panel. The B panel consistently finds more instances of reasoning (of any type) than the M panel and the system. These differences can be explained by the fact that the mental model of the M panel annotators is reflected in the program.

These results are very satisfying from an Artificial Intelligence perspective: they show that, e.g., from the point of view of the B panel, the classification expressed by the M panel and that of the system are *indistinguishable*.

3 Conclusions

We have described the design and implementation of a prototype that analyzes and classifies transcripts of interviews collected during a cognitive science experiment that concerned assessing reasoning bias in lateral-brain damage patients. Our Prolog-based software takes a static description of reasoning rules and matches them on patients' transcripts. Hence, patients' utterances were classified as instances of categorization, prediction and explanation (abduction) based on surface linguistic cues. The agreement between our automatic classifier and human annotators is measured. The agreement is statistically significant, thus showing that the classification can be performed in an automatic fashion. The statistical results support our claim that our software can be safely applied to automate the analysis of experimental results of the type described earlier. Our program can be useful as a provider of second opinions to reveal possible overlooks or mistakes in the diagnostic analysis.

From a Cognitive science point of view, our project may be considered limited by the fact that it can analyze only verbal (transcribed) responses to experiments. Vice versa, from an A.I. point of view we can see that the pattern matching mechanism, though rather basic vis-à-vis current natural language processing techniques is implemented fairly elegantly and efficiently in Prolog.

We are currently working to incorporate such techniques, (e.g., regular expressions) into our program. It would be interesting to apply our classifier to the transcripts of the experiment in [3] since their experiment seems within the reach of the techniques we have employed. Another promising direction of research consist in attaching to the token words some *semantics* obtained by automated reference to Wordnet⁷.

References

- [1] Jean Carletta, 'Assessing agreement on classification tasks: the kappa statistic', *Computational Linguistics*, **22(2)**, 249–254, (1996).
- [2] J. Cohen, 'A coefficient of agreement for nominal scales', *Educational and Psychological Measurement*, **20(1)**, 37–46, (1960).
- [3] Katiuscia Sacco, Monica Bucciarelli, and Mauro Adenzato, 'Mental models and the meaning of connectives: A study on children,adolescents and adults', *Proc. of the XXIIIth Conf. of the Cognitive Science Society*, 875–880, (2001).

⁷ <http://wordnet.princeton.edu>

A Semantics for Active Logic

Mikael Asker and Jacek Malec¹

1 Introduction

Active logics [2] is a family of logics intended to model awareness of resource limitations of the reasoning process. Active logics can be used for reasoning *in time*, handling contradictions and for introspection. Active logics are usually defined syntactically, as axiomatic theories, but until recently have had no formally defined semantics.

A major property of a logic capturing common intuitions about resource-awareness is *paraconsistency*. Speaking informally, this property guarantees that presence of a contradictory pair of sentences in a theory does not necessarily lead to inference of arbitrary consequences, i.e., that the logic is not *explosive*.

The first and so far the only serious attempt to build a semantics for active logics is *active consequence* from [1], presented briefly below. However, it has been recently shown in [3] that active consequence is indeed explosive.

The active consequence relation can be modified in order to escape explosiveness. In this paper we describe our attempt to constrain the active consequence relation so that an active logic based on it becomes paraconsistent.

2 Active consequence

Active consequence was defined in [1] as an attempt to create a semantics for a simple active logic. The language \mathcal{L} is a sorted first-order language defined in two parts: a propositional language \mathcal{L}_w to express facts about the world, and a first order language \mathcal{L}_a used to express facts about the agent and its beliefs. We will use $Sn_{\mathcal{L}}$ to denote the set of all sentences of a language \mathcal{L} .

\mathcal{L}_w is built around a set of sentence symbols, the propositional connectives \neg and \rightarrow and parentheses. The semantics of \mathcal{L}_w is defined by truth assignments. An \mathcal{L}_w -interpretation h is a function $h : Sn_{\mathcal{L}_w} \rightarrow \{\top, \perp\}$. \mathcal{L}_a is built around a set of constant symbols that can refer to points in time, to sentences in \mathcal{L}_w and in \mathcal{L}_a itself, the predicates *now*, *contra* and *bel* and the propositional connective \neg . The intended meanings of *now*, *contra* and *bel* are to indicate the current time, the existence of a direct contradiction at some time and that the agent has a belief at some time, respectively. The semantics of \mathcal{L}_a is based on \mathcal{L}_a -structures H_t^a , which describe the reasoning history of agent a at time t . An \mathcal{L}_a -structure H_t^a can be built from the time sequence $\langle KB_k^a \rangle_{k=0}^t$ of the knowledge bases of agent a up to time t .

The language \mathcal{L} is defined so that $Sn_{\mathcal{L}} = Sn_{\mathcal{L}_w} \cup Sn_{\mathcal{L}_a}$. Its semantics is based on active structures $M_t^a = \langle h_t, H_t^a \rangle$, consisting of one \mathcal{L}_w -interpretation h_t and one \mathcal{L}_a -structure H_t^a . More details about the languages \mathcal{L} , \mathcal{L}_w and \mathcal{L}_a can be found in [1] or [3].

The basic idea of *active consequence* is that the agent perceives only a part of its database, and thus only some of the direct contradictions in the database are visible at a given instant. Active consequence models this by watching the database through a *perception function*, which maps the \mathcal{L} -sentences in the database to another language \mathcal{L}' . PER_t is the set of all possible perception functions at time t . By adding extra indexes to the sentence symbols S_j in the \mathcal{L}_w formulas in the database, they are mapped to symbols S_j^i in another language \mathcal{L}'_w . For a direct contradiction to be visible, the sentence symbols in both of the formulas involved in the contradiction must be mapped to the same indexes. If they are not, then the \mathcal{L}'_w -symbols are different and there is no contradiction, so we can hide a direct contradiction by mapping the symbols in the \mathcal{L} -formulas involved to different extra indexes. The basic idea of active consequence is now that one database follows from another, if there are ways to perceive the databases (perception functions) for which there is classical consequence between the perceptions. Unfortunately, in [3] it was proved that active consequence is explosive.

3 Local consequence

An analysis of the explosivity proof in [3] shows that one important problem is that we can map the same variable to two values at the same time. This is what causes the explosion, and in this sense, the perceptions functions in active consequence are too general.

One idea about how to cure the problems is to restrict the choice of perception functions. One reasonable way to restrict the choice of perception functions is to take subsets of the lhs and rhs formulas. If we require the subsets to be consistent, then classical logical consequence between the perceptions is meaningful in the sense that models do exist. What we want to do is to take a consistent subset of the lhs and take the subset of the rhs which contains the new formulas. Technically, this can be realized inside the framework of active consequence, by mapping the symbols in a formula to index 1 iff the formula is a member of the subset. Otherwise its terms are mapped to unique indexes different from 1 to avoid restricting the model set. One more requirement which turns out to be necessary is that the old formulas on the right side are mapped to the same indexes as on the left side.

Definition 1 (Local perception function pair) Let Σ and $\Theta \subseteq Sn_{\mathcal{L}}$ and denote with Γ_1 the set $\Sigma \cap Sn_{\mathcal{L}_w}$ and with Γ_2 the set $\Theta \cap Sn_{\mathcal{L}_w}$. Order the formulas in Γ_1 and Γ_2 lexicographically as $\varphi_{11}, \varphi_{12}, \dots, \varphi_{1m_1}$ and $\varphi_{21}, \varphi_{22}, \dots, \varphi_{2m_2}$, respectively. Let $\langle S_{j_{11}}, S_{j_{12}}, \dots, S_{j_{1n_1}} \rangle$ and $\langle S_{j_{21}}, S_{j_{22}}, \dots, S_{j_{2n_2}} \rangle$ be the finite sequences of all sentence-symbol tokens occurring in the strings $\varphi_{11}\varphi_{12}\dots\varphi_{1m_1}$ and $\varphi_{21}\varphi_{22}\dots\varphi_{2m_2}$, the concatenated formulas. For each k_1 in $\{1, \dots, n_1\}$ or k_2 in $\{1, \dots, n_2\}$, there is a corresponding formula φ_{1l} , $1 \leq l \leq m_1$ or φ_{2l} , $1 \leq l \leq m_2$, to which the sentence symbol $S_{j_{1k_1}}$ or $S_{j_{2k_2}}$ belongs.

¹ Department of Computer Science, Lund University, Sweden, email: jacek@cs.lth.se, URL: <http://ai.cs.lth.se>

For each $\Delta \subseteq \Gamma_1$, we can define a sequence of positive integers $\langle i_{11}, i_{12}, \dots \rangle$ by $i_{1k} = 1$ if $\varphi_{1l} \in \Delta$, where φ_{1l} is the formula in Γ_1 containing $S_{j_{1k}}$ or if $k > n_1$, $k+1$ otherwise.

We also define the sequence $\langle i_{21}, i_{22}, \dots \rangle$ of positive integers as $i_{2k} = 1$ if $\varphi_{2l} \in \Gamma_2 \setminus \Gamma_1$, where φ_{2l} is the formula in Γ_2 containing $S_{j_{2k}}$ or if $k > n_2$, otherwise $\varphi_{2l} \in \Gamma_2 \cap \Gamma_1 \subseteq \Gamma_1$ - use the same value as that used for φ_{2l} in the i_{1k} sequence.

The sequences $\langle i_{11}, i_{12}, \dots \rangle$ and $\langle i_{21}, i_{22}, \dots \rangle$ above together define a pair of perception functions ($\text{pert}_t, \text{pert}_{t+1}$).

The set $\text{PER}_{lt}(\Sigma, \Theta)$ is the set of perception function pairs defined as above for all consistent subsets $\Delta \subseteq \Sigma \cap \text{Sn}_{\mathcal{L}_w}$. This defines the function $\text{PER}_{lt} : \Sigma_t^\omega \times \mathcal{P}(\text{Sn}_{\mathcal{L}}) \rightarrow \mathcal{P}(\text{PER}_t) \times \mathcal{P}(\text{PER}_{t+1})$ which generates the set of possible local perception function pairs from the sets of premisses and results.

Definition 2 (Local consequence) Let $\Sigma \in \Sigma_t^\omega$ be t -weakly consistent, and let $\Theta \subseteq \text{Sn}_{\mathcal{L}}$ be arbitrary. We say that Θ is a 1-step local consequence of Σ at time t , written $\Sigma \models_{l1} \Theta$, if and only if

$$\begin{aligned} \exists (\text{pert}_t, \text{pert}_{t+1}) \in \text{PER}_{lt} : \forall a : [KB_t^a = \Sigma \rightarrow \\ [H_{t+1}^a \models (\text{pert}_{t+1}(\Theta) \cap \text{Sn}_{\mathcal{L}_a}) \wedge M(\text{pert}_t(\Sigma) \cap \text{Sn}_{\mathcal{L}'_w}) \neq \emptyset \wedge \\ (\text{pert}_t(\Sigma) \cap \text{Sn}_{\mathcal{L}'_w}) \models (\text{pert}_{t+1}(\Theta) \cap \text{Sn}_{\mathcal{L}'_w})]] \end{aligned}$$

n -step local consequence is defined recursively:

$$\Sigma \models_{ln} \Theta \Leftrightarrow \exists \Gamma \subseteq \text{Sn}_{\mathcal{L}} : \Sigma \models_{l(n-1)} \Gamma \wedge \Gamma \models_{l1} \Theta$$

Finally, local consequence in general is defined as:

$$\Sigma \models_l \Theta \Leftrightarrow \exists n \in \mathbb{N} : \Sigma \models_{ln} \Theta$$

The following theorems show that Definition 2 is correct, in the sense that the new consequence relation has the desired properties:

Theorem 1 (Local consequence is local)

$$\begin{aligned} \forall (\Sigma, \Theta) \in \Sigma_t^\omega \times \mathcal{P}(\text{Sn}_{\mathcal{L}}) : \Sigma \models_{l1} \Theta \Rightarrow \\ \exists \Sigma' \subseteq \Sigma \cap \text{Sn}_{\mathcal{L}_w} : M(\Sigma') \neq \emptyset \wedge \Sigma' \models (\Theta \setminus \Sigma) \cap \text{Sn}_{\mathcal{L}_w} \\ \forall (\Sigma, \Theta) \in \mathcal{P}(\text{Sn}_{\mathcal{L}_w}) \times \mathcal{P}(\text{Sn}_{\mathcal{L}_w}) : \\ (\exists \Sigma' \subseteq \Sigma : M(\Sigma') \neq \emptyset \wedge \Sigma' \models \Theta \setminus \Sigma) \Rightarrow \Sigma \models_{l1} \Theta \end{aligned}$$

Theorem 2 (Local consequence is active) $\models_l \subset \models_a$.

Theorem 3 (Local consequence is paraconsistent)

$$\neg \forall \Sigma \in \Sigma_t^\omega : \forall (\varphi, \psi) \in \text{Sn}_{\mathcal{L}_w}^2 : \{\varphi, \neg\varphi\} \subseteq \Sigma \Rightarrow \Sigma \models_l \Sigma \cup \{\psi\}$$

4 Relation to classical logic

Theorem 4 (The world part of local consequence is classical)

$$\forall (\Sigma, \Theta) \in \Sigma_t^\omega \times \mathcal{P}(\text{Sn}_{\mathcal{L}}) : \Sigma \models_{l1} \Theta \Rightarrow \Sigma \cap \text{Sn}_{\mathcal{L}_w} \models \Theta \cap \text{Sn}_{\mathcal{L}_w}$$

Next, we show that for strongly consistent databases, \models_l is equivalent to classical consequence:

Theorem 5

$$\forall (\Sigma, \Theta) \in [\mathcal{P}(\text{Sn}_{\mathcal{L}_w})]^2 : M(\Sigma) \neq \emptyset \Rightarrow (\Sigma \models_{l1} \Theta \Leftrightarrow \Sigma \models \Theta)$$

Let \models_{amp} be the consequence relation which is generated by the *active modus ponens* inference rule, the active logics variant of the traditional modus ponens rule:

$$\frac{t : \varphi, \varphi \rightarrow \psi}{t + 1 : \psi}, \quad \text{where } \varphi, \psi \in \text{Sn}_{\mathcal{L}_w} \quad (1)$$

Active modus ponens is certainly a rule which we want to be locally sound. But unfortunately, because $\{\neg(S_1 \rightarrow S_1), (\neg(S_1 \rightarrow S_1)) \rightarrow S_2\} \not\models_l \{S_2\}$ it isn't, $\models_{amp} \setminus \models_l \neq \emptyset$! We want to handle active modus ponens, but still be paraconsistent. So we have to find a consequence relation \models_x “between” \models_l and \models_a , with $\models_l \subset \models_x$ to include active modus ponens, but with $\models_x \subset \models_a$ to remain paraconsistent.

The part of \models_{amp} that is inside \models_l is the part where the premisses have models and the reasoning is “truly meaningful”. The part of \models_{amp} that is outside \models_l , the *semantic residual*, is exactly the part where the premisses are inconsistent. We interpret the residual as a reflection of artifacts in the proof theory. The residual models the proof theory when it is doing “meaningless” reasoning with inconsistent premisses.

We believe that there is a reason for why the proof theory does this kind of “meaningless” reasoning, and that the reason is that the inference rule has no way to find out that the reasoning is “meaningless”. Inference rules are syntactic operations which can be implemented efficiently with simple pattern matching, and are “too stupid” to discover inconsistency. Computationally, discovering inconsistency is expensive. The residual part is present also in classical logic, it is just that nobody has bothered about it before.

The residual part must be included in a semantic consequence relation to achieve soundness, and because active modus ponens is *active* sound, we can include the residual in the semantic consequence relation and still have that relation being a subset of active consequence. Now the utility of being a subset of active consequence as proved in Theorem 2 becomes apparent. That property enables us to go outside \models_l to other parts of \models_a to cover the residual parts of the rules.

5 Conclusions

The work described in this paper directly builds on the earlier work on Active Logics: logics aware of passage of time and the computational cost of inference.

Active consequence in its original form is not very useful as a semantics for active logics itself, because it is explosive. But this text shows that it is indirectly useful, in that can be used as a framework for defining other consequence relations which are subsets of active consequence, by restricting the choice of perception functions.

One such subset is local consequence, which was defined and shown to be paraconsistent. What we aim at as our final semantic consequence relation is probably a part of local consequence, plus some semantic residuals to compensate for the stupidity of other inference rules.

REFERENCES

- [1] Michael L. Anderson, Walid Gomaa, John Grant, and Don Perlin, ‘On the reasoning of real-world agents: Toward a semantics for active logic’, in *Proceedings of the 7th Annual Symposium on the Logical Formalization of Commonsense Reasoning*, Corfu, Greece, (2005).
- [2] Jennifer Elgot-Drapkin, Sarit Kraus, Michael Miller, Madhura Nirke, and Donald Perlin, ‘Active logics: A unified formal approach to episodic reasoning’, Technical Report CS-TR-4072, Department of Computer Science, University of Maryland, College Park, Maryland, (1999).
- [3] Johan Hovold, *On a Semantics for Active Logic*, Master’s thesis, Department of Computer Science, Lund University, Lund, Sweden, 2005. Available at <http://ai.cs.lth.se/xj/JohanHovold/finalreport.pdf>.

An alternative inference for Qualitative Choice Logic

Salem Benferhat and Daniel Le Berre and Karima Sedki¹

Abstract. *Qualitative Choice Logic* adds to classical propositional logic a new connective, called *ordered disjunction*, used to express preferences between alternatives. We present an alternative inference relation for the *QCL* language that overcomes some *QCL* limitations.

1 Introduction

Recently, a new logic for representing choices and preferences has been proposed [1]. This logic is called *Qualitative Choice Logic (QCL)* and is an extension of propositional logic. The non-standard part of *QCL* logic is a new logical connective $\vec{\times}$, called *Ordered disjunction*, which is fully embedded in the logical language. Intuitively, if A and B are propositional formulas then $A \vec{\times} B$ means: “if possible A, but if A is impossible then at least B”. *QCL* logic is very useful to represent preferences. However, its inference tool is less satisfactory. In fact, the way the negation is handled in *QCL* logic is not fully satisfactory. In *QCL* when a negation is used on a *QCL* formula with ordered disjunction, that negated *QCL* formula is logically equivalent to a propositional formula obtained by replacing the ordered disjunction ($\vec{\times}$) by the propositional disjunction (\vee). We propose to equip the *QCL* framework with a new inference relation that overcomes the *QCL* original inference relation.

2 Basic Choice Formulas (BCF)

Let $PROP_{PS}$ be a propositional language based on a set of propositional symbols PS . Basic choice formulas are ordered disjunctions of propositional formulas. They propose a simple way to order available alternatives. The operator $\vec{\times}$ is associative. Given propositional formulas a_1, a_2, \dots, a_n , the formula $a_1 \vec{\times} a_2 \vec{\times} \dots \vec{\times} a_n$ is used to express an ordered list of alternatives: some a_i must be true, preferably a_1 , but if this is not possible then a_2 , if this is not possible a_3 , etc. The language composed of *basic choice formulas* and propositional formulas is denoted by BCF_{PS} , and defined by:

Definition 1 *The set BCF_{PS} of basic choice formulas is the smallest set of words defined inductively as follow:*

1. If $\phi \in PROP_{PS}$ then $\phi \in BCF_{PS}$
2. If $\phi, \psi \in BCF_{PS}$ then $(\phi \vec{\times} \psi) \in BCF_{PS}$
3. Every *basic choice formula* is obtained by applying the two rules above a finite number of times.

¹ CRIL-CNRS FRE 2499, Université d'Artois, France, email: {benferhat,leberre,sedki}@cril.univ-artois.fr

The semantics of BCF is based on the degree of satisfaction of a formula in a particular model. As in standard propositional logic, an interpretation I is an assignment of the classical truth values T,F to the atoms in PS .

I will be represented by the set of its satisfied literals.

Definition 2 [1]

- Let $\phi = a_1 \vec{\times} a_2 \vec{\times} \dots \vec{\times} a_n \in BCF_{PS}$.
 $I \models_k a_1 \vec{\times} a_2 \vec{\times} \dots \vec{\times} a_n$ iff $I \models a_1 \vee a_2 \vee \dots \vee a_n$ and $k = \min(j \mid I \models a_j)$.
- Let $\phi \in PROP_{PS}$. $I \models_1 \phi$ iff $I \models \phi$.

Namely, a basic choice formula $a_1 \vec{\times} a_2 \vec{\times} \dots \vec{\times} a_n$ is satisfied to a degree k by an interpretation I if I satisfies a_k but fails to satisfy a_i for all $1 \leq i < k$.

The language of *basic choice formulas* has strong relationships with possibility theory, in particular with guaranteed possibility distributions, (see [1] for more details).

The following introduces the notion of equivalences between *basic choice formulas*.

Definition 3 [1] *Two basic choice formulas ϕ and ψ are said to be equivalent, denoted simply by $\phi \equiv \psi$, if for all interpretation I , and integer k we have $I \models_k \phi$ iff $I \models_k \psi$.*

In the following, K is a set of propositional formulas which represents knowledge or integrity constraints, and T is a set of preferences. To define the inference relation between $(K \cup T)$ and a propositional formula ϕ , we first need to define the notion of models and preferred models.

Definition 4 *Let K be a set of propositional formulas and T be a set of preferences. An interpretation I is a model of $K \cup T$ iff*

1. *I satisfies K, and*
2. *I satisfies each formula of T to some degree (i.e., $\forall \phi \in T, \exists k$ such that $I \models_k \phi$).*

The satisfaction degree on formulas helps us to determine preferred models. There are different ways of doing this. In [1] a lexicographic ordering on models, based on the number of formulas satisfied to a particular degree is used. The lexicographic ordering is defined as follows:

Definition 5 *Let $M^k(T)$ denote the subset of formulas of T satisfied by a model M to a degree k . A model M_1 is $K \cup T$ -preferred over a model M_2 if there is a k such that $|M_1^k(T)| > |M_2^k(T)|$ and for all $j < k$: $|M_1^j(T)| = |M_2^j(T)|$. M is a preferred model of $K \cup T$ iff M is maximally $(K \cup T)$ -preferred.*

Definition 6 Let K be a set of propositional formulas and T be a set of preferences, and A be a propositional formula of $PROP_{PS}$. $K \cup T \not\vdash A$ iff A is satisfied in all preferred models of $K \cup T$.

Example 1 As an example consider $K = \emptyset$ and $T = \{\text{Air-France} \vec{x} \text{EasyJet}\}$. We have three models $\{\text{Air-France}\}$, $\{\text{Air-France}, \text{EasyJet}\}$, $\{\text{EasyJet}\}$ with respective satisfaction degree 1, 1, 2. This means that $\{\text{Air-France}\}$ and $\{\text{Air-France}, \text{EasyJet}\}$ are maximally preferred and we have $\text{Air-France} \vec{x} \text{EasyJet} \not\sim \text{Air-France}$.

After building examples of general QCL formulas, we noticed that the QCL logic has two main limitations:

- **The double negation of a QCL formula does not preserve the initial formula, namely $\neg\neg\phi \not\equiv \phi$.** For instance, let $\phi = (a \vec{x} b) \vee \neg c$, where a, b, c are three independent propositional symbols. Then, in QCL framework
 $\neg\neg\phi \equiv \neg\neg((a \vec{x} b) \vee \neg c)$
we have: $\equiv \neg((\neg a \wedge \neg b) \wedge c)$
 $\equiv a \vee b \vee \neg c$.
- **Implication is not well handled.** For instance, the QCL framework does not make distinction between the following three rules:
 $(\text{AirFrance} \vec{x} \text{EasyJet}) \Rightarrow \text{HotelPackage}$,
 $(\text{EasyJet} \vec{x} \text{AirFrance}) \Rightarrow \text{HotelPackage}$,
 $(\text{AirFrance} \vee \text{EasyJet}) \Rightarrow \text{HotelPackage}$.

3 An alternative inference relation for the QCL language

Let QCL_{PS} be $PROP_{PS}$ augmented by the binary connective \vec{x} . The main limitations we identified for QCL_{PS} are the consequences of using the negation on formulas containing ordered disjunctions. It is not easy to give a precise meaning of negative preferences: what does really mean “I do not prefer flying with Air France than EasyJet”? Most of existing works on logic-based representation of preference ignore or does not allow the representation of negated preferences [2, 3, 4]. In our mind, preserving the double negation and the use of implications are desirable properties.

To define our new inference relation on QCL_{PS} , we will follow the same steps as used in [1], namely:

1. First, we transform the set of *general choice formulas* into a set of *basic choice formulas*,
2. Then use Definition 2, 4, 5 to draw the set of consequences.

The transformation of $\phi \otimes \psi$ (with $\otimes = \vee, \wedge, \vec{\otimes}$ a binary connective from the QCL logic) is equivalent to $\phi' \otimes \psi'$ where ϕ' and ψ' are the transformation of ϕ and ψ .

The transformation should also respect De Morgan’s law.

For the specific cases of a conjunction of BCF or the negation of a BCF , we use the following transformation rules.

Let $\phi = a_1 \vec{x} a_2 \vec{x} \dots \vec{x} a_n$, and $\psi = b_1 \vec{x} b_2 \vec{x} \dots \vec{x} b_m$ such that a_i ’s and b_i ’s are propositional formulas.

- $((a_1 \vec{x} a_2 \vec{x} \dots \vec{x} a_n) \wedge (b_1 \vec{x} b_2 \vec{x} \dots \vec{x} b_m)) \equiv c_1 \vec{x} c_2 \vec{x} \dots \vec{x} c_k$ where $k = \max(m, n)$, and

$$c_i = \begin{cases} (a_i \wedge b_i) & \text{if } i \leq \min(m, n) \\ a_i & \text{if } m \leq i \leq n \\ b_i & \text{if } n \leq i \leq m \end{cases}$$

- $\neg(a_1 \vec{x} a_2 \vec{x} \dots \vec{x} a_n) \equiv \neg a_1 \vec{x} \neg a_2 \vec{x} \dots \vec{x} \neg a_n$.

Example 2

Here is an example of the way disjunctions of BCF are handled in our new inference system:

$$\begin{aligned} \neg\neg\phi &\equiv \neg\neg((a \vec{x} b) \vee \neg c) \\ &\equiv \neg(\neg a \vec{x} \neg b) \wedge c \\ &\equiv a \vec{x} b \vee \neg c \quad \equiv a \vee \neg c \vec{x} b \end{aligned}$$

Hence, contrary to the QCL we do not loose the ordered disjunction in the scope of a negation. A more interesting result arises when considering implications:

$$\text{AirFrance} \vec{x} \text{EasyJet} \Rightarrow \text{FirstClass}$$

$$\equiv (\neg \text{AirFrance} \vee \text{FirstClass}) \vec{x} \neg \text{EasyJet}$$

While:

$$\text{EasyJet} \vec{x} \text{AirFrance} \Rightarrow \text{FirstClass}$$

$$\equiv (\neg \text{EasyJet} \vee \text{FirstClass}) \vec{x} \neg \text{AirFrance}$$

Therefore the order of preferences is very important.

Let us consider finally the example where our knowledge base K contains $\neg \text{EasyJet} \vee \neg \text{AirFrance}$ (1) and T contains the following preferences:

$$\left\{ \begin{array}{l} \text{AirFrance} \vec{x} \text{EasyJet} \Rightarrow \text{HotelPackage}(2) \\ \text{EasyJet} \vec{x} \text{AirFrance} \Rightarrow \neg \text{HotelPackage}(3) \\ \text{AirFrance} \vec{x} \text{EasyJet}(4) \end{array} \right.$$

The following truth table summarizes at which degree of satisfaction is satisfied each preference (1 or 2) with the new inference relation.

AirFrance	EasyJet	Hotel	(1)	(2)	(3)	(4)
F	F	F	1	1	1	-
F	F	T	1	1	1	-
F	T	F	1	1	2	2
F	T	T	1	1	1	2
T	F	F	1	2	1	1
T	F	T	1	1	1	1
T	T	F	-	-	-	1
T	T	T	-	1	1	1

The given $T \cup K$ has one preferred model (bold line), $I = \{\text{AirFrance}, \text{Hotel}\}$, from which we get the expected conclusion $K \cup T \not\sim \text{Hotel}$, not derivable from QCL framework.

4 Conclusion

We presented a limitation of the original QCL inference, resulting from the way negations are handled: negated QCL formulas are equivalent to plain propositional formulas. We thus proposed a new inference framework for the QCL language that overcomes this limit: the new inference scheme preserves preferential information in negated QCL formulas.

REFERENCES

- [1] Gerhard Brewka, Salem Benferhat, and Daniel Le Berre, ‘Qualitative choice logic’, *Artif. Intell.*, **157**(1-2), 203–237, (2004).
- [2] Boutilier, C., Towards a Logic for Qualitative Decision Theory, Proc. Principles of Knowledge Representation and Reasoning, KR-94, 1994, pp 75-86
- [3] J. Lang. van der Torre, E. Weydert, Utilitarian Desires, Autonomous Agents and Multi-Agent Systems 5(3) (2002) 329-363.
- [4] Tan S.-W., Pearl J. (1994) qualitative decision theory. In: Proc. of the 12yh National Conf. on Artificial Intelligence (AAAI-94), Seattle, Wa., July 31 - Aug. 4, 1994, 928-933.

On the Existence of Answer Sets in Normal Extended Logic Programs¹

Martin Caminada² and Chiaki Sakama³

Abstract. One of the serious problems in *answer set programming* is that relatively small pieces of information can cause a total absence of answer sets. To cope with this problem, this paper introduces a class of *normal extended logic programs* which are extended logic programs, whose defeasible rules are comparable to normal defaults in default logic. Under suitable program transformations, we show that every normal extended logic program always yields at least one answer set.

1 Introduction

Answer set programming (ASP) is a declarative programming paradigm which is useful for AI problem solving [1, 5]. In ASP a program is represented as an *extended logic program* whose declarative meaning is given by the *answer set semantics* [4]. However, it is well-known that a program does not always have an answer set. A notorious example is the rule like $p \leftarrow \text{not } p$. The existence of such a “negative loop” – an atom depending on its default negation, can cause a total absence of answer sets, and blocks all useful inferences from other parts of the program.

From the knowledge representation viewpoint, a negative loop is often considered anomalous information. So there would be a good reason that a program including such anomalous information is unusual anyway. The problem, however, is that even programs that do not include any such anomalous information often fail to have an answer set. This is illustrated by, for instance, the “Married John” example from [2]: (a) “John wears something that looks like a wedding ring.” (b) “John parties with his friends until late.” (c) “Someone wearing a wedding ring is usually married.” (d) “A party-animal is usually a bachelor.” (e) “A married person, by its definition, has a spouse.” (f) “A bachelor, by definition, does not have a spouse.” These sentences are represented by the following program:

$$\begin{array}{ll} r \leftarrow & p \leftarrow \\ m \leftarrow r, \text{not } \neg m & b \leftarrow p, \text{not } \neg b \\ hs \leftarrow m & \neg hs \leftarrow b \end{array}$$

This program contains no negative cycle, and encodes given information in a natural way. The program has no answer set, however.

One may consider that interpreting strict (NAF-free) rules as *clauses* in propositional logic and representing them as disjunctive facts would solve the problem. Rewriting the strict rules as disjunctive facts in the above program would yield the following.

$$\begin{array}{ll} r \leftarrow & p \leftarrow \\ m \leftarrow r, \text{not } \neg m & b \leftarrow p, \text{not } \neg b \\ hs \vee \neg m \leftarrow & \neg hs \vee \neg b \leftarrow . \end{array}$$

This program has two answer sets: $\{r, p, m, \neg b, hs\}$ and $\{r, p, b, \neg m, \neg hs\}$. Unfortunately, such rewriting does not work in general. For instance, the following program contains only normal defeasible rules and disjunctive facts but still has no answer set.⁴

$$\begin{array}{ll} \neg a \leftarrow b, \text{not } a, & b \leftarrow a, \text{not } \neg b, \\ c \leftarrow \neg a, \text{not } \neg c, & d \leftarrow c, \text{not } \neg d, \\ a \leftarrow d, \text{not } \neg a, & a \vee b \vee c \leftarrow . \end{array}$$

The above discussion indicates that specifying syntactic restrictions under which an extended logic program is guaranteed to have answer sets is not a trivial task, especially when one also want to preserve a form of strict reasoning.

2 Basic Definitions

A *program* considered in this paper is an *extended logic program* (ELP) [4], which is a finite set of *rules* of the form:

$$c \leftarrow a_1, \dots, a_n, \text{not } b_1, \dots, \text{not } b_m \quad (1)$$

where each c , a_i and b_j is a positive/negative literal and *not* stands for *default negation* or *negation as failure* (NAF). *not* b_j is called an *NAF-literal*. If a is an atom, we identify $\neg \neg a$ with a . The literal c is the *head* and the conjunction $a_1, \dots, a_n, \text{not } b_1, \dots, \text{not } b_m$ is the *body*. The head is nonempty, while the body is possibly empty. For each rule r of the form (1), *head(r)* represents the literal c , and *body⁺(r)* and *body⁻(r)* represent the sets $\{a_1, \dots, a_n\}$ and $\{b_1, \dots, b_m\}$, respectively. A rule is called *strict* if it is of the form:

$$c \leftarrow a_1, \dots, a_n. \quad (2)$$

Otherwise, a rule (1) is called *defeasible*. Given a program P , we use the notation *strict(P)* for the set of all strict rules of P , and *defeasible(P)* for the set of all defeasible rules of P . Clearly, $P = \text{strict}(P) \cup \text{defeasible}(P)$. A program is *NAF-free* if it consists of strict rules only. A program (rule, literal) is *ground* if it contains no variable. Throughout the paper, we handle finite ground programs unless stated otherwise.

The semantics of ELPs is given by the *answer set semantics* [4]. Let Lit be the set of all ground literals in the language of a program. A set $S (\subseteq \text{Lit})$ satisfies a ground rule r if $\text{body}^+(r) \subseteq S$ and $\text{body}^-(r) \cap S = \emptyset$ imply $\text{head}(r) \in S$. S satisfies a program P if

¹ This work has been sponsored by the EU ASPIC project.

² Institute of Information and Computing Sciences, P.O. Box 80 089 3508 TB Utrecht The Netherlands; email: martinc@cs.uu.nl

³ Department of Computer and Communication Sciences, Wakayama University, Wakayama 640-8510, Japan; email: sakama@sys.wakayama-u.ac.jp

⁴ We invite those who claim to have found one to verify the minimality of their solution.

S satisfies every rule in P . Let P be an NAF-free ELP. Then, a set $S(\subseteq \text{Lit})$ is an *answer set* of P if S is a minimal set such that (i) S satisfies every rule from P ; and (ii) if S contains a pair of complementary literals L and $\neg L$, $S = \text{Lit}$. Next, let P be any ELP and $S \subseteq \text{Lit}$. For every rule r of P , the rule $\text{head}(r) \leftarrow \text{body}^+(r)$ is included in the *reduct* P^S if $\text{body}^-(r) \cap S = \emptyset$. Then, S is an *answer set* of P if S is an answer set of P^S . An answer set is *consistent* if it is not *Lit*. A program P is *consistent* if it has a consistent answer set; otherwise P is *inconsistent*. Remark that, by the definition, an ELP P has the answer set *Lit* iff $\text{strict}(P)$ has the answer set *Lit*.

3 Normal Extended Logic Programs

Extended logic programs often fail to have an answer set. A similar problem arises in the context of Reiter's *default logic*. To deal with the problem of the potential non-existence of extensions in default logic, a possible solution is to restrict the syntax of knowledge representation. In [6], for instance, it is shown that a *normal default theory*, in which every default has the form $\frac{\alpha:\beta}{\beta}$, always yields at least one extension. In spite of their restricted syntax, normal default theories are useful to encode a large class of commonsense knowledge. An interesting question is then whether such an approach would also be feasible for logic programming. That is, can we state some possible restrictions on the syntax and content of an ELP, under which the existence of answer sets is guaranteed?

Analogously to default logic, one possible solution would be to restrict the use of NAF in defeasible rules. That is, we restrict default negation only to occur for a literal that is the opposite of the head of the same rule. More precisely, a defeasible rule of the form:

$$c \leftarrow a_1, \dots, a_n, \text{not } \neg c \quad (3)$$

is called a *normal rule*. There is a good reason to call this type of rules normal. In fact, according to [4], the above normal rule is essentially the same as a normal default of the form: $a_1 \wedge \dots \wedge a_n : c / c$.

Unfortunately, the mere restriction that all defeasible rules should be normal rules is not enough to guarantee the existence of answer sets. The "Married John" example illustrated in Section 1 is a counter-example of this restriction. One possible diagnosis of the "Married John" example is that, apparently, some information is missing. From our commonsense knowledge, we know that someone without a spouse is not married ($\neg m \leftarrow \neg hs$) and that someone who has a spouse is not a bachelor ($\neg b \leftarrow hs$). Notice that these two rules are actually contrapositive forms of the existing rules $hs \leftarrow m$ and $\neg hs \leftarrow b$. Adding these rules yields the following logic program.

$$\begin{array}{ll} r \leftarrow & hs \leftarrow m \\ p \leftarrow & \neg m \leftarrow \neg hs \\ m \leftarrow r, \text{not } \neg m & \neg hs \leftarrow b \\ b \leftarrow p, \text{not } \neg b & \neg b \leftarrow hs. \end{array}$$

The above program has two answer sets: $\{r, p, m, hs\}$ and $\{r, p, b, \neg hs\}$. This outcome can be seen as more desirable than the outcome of the original program, where no answer set exists.

Nevertheless, contraposition (or even *transposition*, as introduced in [2]) may not be enough to guarantee the existence of answer sets. Consider the following program:

$$\begin{array}{ll} a \leftarrow & b \leftarrow a, \text{not } \neg b \\ c \leftarrow b & \neg b \leftarrow c. \end{array}$$

In this program, all defeasible rules are normal, but even when one adds the rules $\neg b \leftarrow \neg c$ and $\neg c \leftarrow b$ (which makes the set of strict rules closed under contraposition) the program still does not yield any answer sets. It indicates that in order to guarantee the existence of answer sets, additional requirements are necessary.

Definition 3.1 (transpositive, transitive, antecedent-cleaned)

Let s_1 and s_2 be strict rules. We say that s_2 is a *transpositive* version of s_1 iff:

$$s_1 = c \leftarrow a_1, \dots, a_n \text{ and}$$

$$s_2 = \neg a_i \leftarrow a_1, \dots, a_{i-1}, \neg c, a_{i+1}, \dots, a_n \text{ for some } 1 \leq i \leq n.$$

Let s_1, s_2 and s_3 be strict rules. We say that s_3 is a *transitive* version of s_1 and s_2 iff:

$$s_1 = c \leftarrow a_1, \dots, a_n,$$

$$s_2 = a_i \leftarrow b_1, \dots, b_m \text{ for some } 1 \leq i \leq n, \text{ and}$$

$$s_3 = c \leftarrow a_1, \dots, a_{i-1}, b_1, \dots, b_m, a_{i+1}, \dots, a_n.$$

Let s_1 and s_2 be strict rules. We say that s_2 is an *antecedent cleaned* version of s_1 iff:

$$s_1 = \neg a_i \leftarrow a_1, \dots, a_i, \dots, a_n \text{ and}$$

$$s_2 = \neg a_i \leftarrow a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n.$$

The intuition behind transposition can be illustrated by translating a strict rule $c \leftarrow a_1, \dots, a_n$ to a material implication $c \subset a_1 \wedge \dots \wedge a_n$. This implication is logically equivalent to $\neg a_i \subset a_1 \wedge \dots \wedge a_{i-1} \wedge \neg c \wedge a_{i+1} \wedge \dots \wedge a_n$, which is again translated to $\neg a_i \leftarrow a_1, \dots, a_{i-1}, \neg c, a_{i+1}, \dots, a_n$. Notice that, when $n = 1$, transposition coincides with classical contraposition. Transitivity and antecedent cleaning can be analyzed similarly.

Definition 3.2 (closed) Let S be a set of strict rules. Then,

- (i) S is *closed under transposition* iff for each rule s_1 in S , a rule s_2 is in S if s_2 is a transpositive version of s_1 .
- (ii) S is *closed under transitivity* iff for each rule s_1 and s_2 in S , a rule s_3 is in S if s_3 is a transitive version of s_1 and s_2 .
- (iii) S is *closed under antecedent cleaning* iff for each rule s_1 in S , a rule s_2 is in S if s_2 is an antecedent cleaned version of s_1 .

Definition 3.3 (normal ELP) A program P is called a *normal extended logic program* (normal ELP, for short) iff:

- (1) $\text{strict}(P)$ is closed under transposition, transitivity and antecedent cleaning, and
- (2) $\text{defeasible}(P)$ consists of normal rules only.

Theorem 1 Any normal ELP P has at least one answer set.

Finally, it is worth mentioning that in the context of the well-founded semantics a semi-normal extended logic program, which consists of a set of defeasible rules of the form $c \leftarrow a_1, \dots, a_n$, $\text{not } b_1, \dots, \text{not } b_m$, $\text{not } \neg c$ and a consistent set of strict rules closed under transposition (but not necessarily under transitivity and antecedent cleaning) always has a consistent well-founded model [3]. In contrast to the present work, the study shows that for well-founded semantics, weaker conditions on the content of the program apply.

REFERENCES

- [1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2002.
- [2] M. Caminada and L. Amgoud. An axiomatic account of formal argumentation. In: *Proceedings of the 20th National Conference of Artificial Intelligence*, MIT Press, 2005.
- [3] M. Caminada. Well-founded semantics for semi-normal extended logic programs. In: *Proceedings of the 11th International Workshop of Non-monotonic Reasoning, special session on answer set programming*, forthcoming, 2006.
- [4] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–385, 1991.
- [5] V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence* 138:39–54, 2002.
- [6] R. Reiter. A logic for default reasoning, *Artificial Intelligence* 13:81–132, 1980.

Smoothed particle filtering for dynamic Bayesian networks

Theodore Charitos¹

Abstract. Particle filtering (PF) for dynamic Bayesian networks (DBNs) with discrete-state spaces includes a resampling step which concentrates samples according to their relative weight in regions of interest of the state-space. We propose a more systematic approach than resampling based on regularisation (smoothing) of the empirical distribution associated with the samples, using the kernel method. We show in our experiments that the smoothed particle filtering (SPF) leads to more accurate estimates than the PF.

1 Introduction

Let $\mathbf{V}_n = (V_n^1, \dots, V_n^m)$, $m \geq 2$, denote the set of variables at time step n . Then, a time-invariant DBN is a tuple (B_1, B_2) , where B_1 is a Bayesian network that represents the prior distribution for the variables at the first time step \mathbf{V}_1 , and B_2 defines the *transition model* in two consecutive time steps, so that for every $n \geq 2$

$$p(\mathbf{V}_n \mid \mathbf{V}_{n-1}) = \prod_{j=1}^m p(V_n^j \mid \pi(V_n^j))$$

where $\pi(V_n^j)$ denotes the set of parents of V_n^j , for $j = 1, \dots, m$. We assume that the set \mathbf{V}_n can be split in two mutually exclusive and collectively exhaustive sets $\mathbf{X}_n = (X_n^1, \dots, X_n^s)$, $\mathbf{Y}_n = (Y_n^1, \dots, Y_n^{m-s})$, where \mathbf{X}_n and \mathbf{Y}_n represent the hidden and observable variables per time step respectively. We use the term *observation model* to denote the probability to observe an instantiation of values \mathbf{y}_n for \mathbf{Y}_n given an instantiation of values \mathbf{x}_n for \mathbf{X}_n . We also denote by $\mathbf{y}_{1:k} \triangleq \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\}$ the observations up to and including time step k .

Monitoring a DBN amounts to compute the probability distribution of the hidden state at time step n given the observations, that is, $p(\mathbf{x}_n \mid \mathbf{y}_{1:n})$. For this task, Murphy [6] introduced the interface algorithm that exploits efficiently the *forward interface* \mathbf{FI}_n , which is the set of variables at time step n that affect some variables at time step $n+1$ directly. However, the computational complexity of the interface algorithm is in every time step exponential in the number of hidden variables and hence exact monitoring can be prohibitive [4], [6]. A way to handle these problems is to use sequential Monte Carlo methods that are easy to implement, work on almost any kind of DBNs and with a large number of samples are guaranteed to provide the exact answer [1], [2], [5].

2 Particle filtering

Let us assume that we are able to sample N independent and identically distributed random samples $\{\mathbf{x}_n^{(i)}; i = 1, \dots, N\}$ according

¹ Department of Information and Computing Sciences, Utrecht University
email: theodore@cs.uu.nl. This research was supported by NWO.

-
- $\omega_n^{(i)} = 1$
 - $\mathbf{x}_n^{(i)}$ is empty
 - for each variable j in a topological order
 - let \mathbf{u} be the value of $\pi(V_n^j)$ in $(\mathbf{x}_{n-1}^{(i)}, \mathbf{x}_n^{(i)})$
 - if $V_n^j \in \mathbf{X}_n$
 - sample $v_n^j \sim p(V_n^j \mid \pi(V_n^j))$
 - set $\mathbf{x}_n^{(i)} = \{\mathbf{x}_n^{(i)}, v_n^j\}$
 - else
 - set v_n^j to be the value of $V_n^j \in \mathbf{Y}_n$
 - $\omega_n^{(i)} = \omega_n^{(i)} \times p(v_n^j \mid \mathbf{u})$
 - Return $(\mathbf{x}_n^{(i)}, \omega_n^{(i)})$
-

Figure 1: Pseudocode for sampling in DBNs.

to $p(\mathbf{X}_n \mid \mathbf{y}_{1:n})$. Then, an empirical estimate of this distribution is given by

$$p(\mathbf{x}_n \mid \mathbf{y}_{1:n}) \approx \frac{1}{N} \sum_{i=1}^N \delta_{\mathbf{x}_n^{(i)}}(d\mathbf{x}_n)$$

where $\delta(d\cdot)$ denotes the Dirac delta function. Typically, we cannot sample efficiently from the posterior distribution $p(\mathbf{X}_n \mid \mathbf{y}_{1:n})$, so instead we sample from a proposal or importance distribution $q(x)$ and weight the samples according to

$$\omega_n^{(i)} \propto \frac{p(\mathbf{x}_n^{(i)} \mid \mathbf{x}_{n-1}^{(i)}, \mathbf{y}_n)}{q(\mathbf{x}_n^{(i)} \mid \mathbf{x}_{n-1}^{(i)}, \mathbf{y}_n)}$$

to obtain the following mass approximation of $p(\mathbf{x}_n \mid \mathbf{y}_{1:n})$

$$p(\mathbf{x}_n \mid \mathbf{y}_{1:n}) \approx \sum_{i=1}^N \tilde{\omega}_n^{(i)} \delta_{\mathbf{x}_n^{(i)}}(d\mathbf{x}_n) \quad (1)$$

where $\tilde{\omega}_n^{(i)}$ is the normalised weight. For DBNs, the generation of a new sample is focused on the Bayesian network constructed on the variables $\mathbf{FI}_{n-1} \cup \mathbf{V}_n$, that represents the transition model B_2 . A pseudocode of this scheme is shown in Figure 1. The PF for monitoring DBNs consists of two consecutive steps: sampling and resampling. Schematically, for $i = 1, \dots, N$, the PF works according to

$$\{\mathbf{x}_n^{(i)}, \tilde{\omega}_n^{(i)}\} \longrightarrow \{\mathbf{x}_n^{(i')}, N^{-1}\} \longrightarrow \{\mathbf{x}_{n+1}^{(i)}, \tilde{\omega}_{n+1}^{(i)}\}$$

The success of the PF depends on whether the Dirac-point mass approximation provides an adequate representation of the posterior distribution. In the resampling step, any particular sample with a high weight will be duplicated many times. As a result, the cloud of samples may eventually collapse to a single sample. This problem is more evident if there is no system noise or the observation noise has very small variance [2]. More refined approaches such as kernel smoothing [7] can help surmount this problem.

3 Smoothed particle filtering

A smoothing scheme was proposed in [4], where the idea was to smooth the joint probability distribution of the hidden state at each time step by adding a smoothing parameter α_o to all entries in the state-space that are consistent with the observations. Sampling from the smoothed probability distribution was then performed in a two-phase process that requires the computation of a constant M which represents the total number of states consistent with the observations. Although simple to implement, this scheme has the disadvantage that the computation of M in general is $\#$ -P hard, while the cost of computing M can be as high as doing exact inference [6, pp. 89].

We propose a smoothing scheme where instead of the joint probability distribution of the hidden state we focus on the marginal probability distribution of the variables that represent the hidden state after inference has been performed. More precisely, from (1) the probability of the hidden variable X_n^j for x_h is

$$p_h = p(X_n^j = x_h \mid \mathbf{y}_{1:n}) \approx \sum_{i:X_n^{j(i)}=x_h} \tilde{\omega}_n^{(i)} \quad (2)$$

As we already argued, the PF can estimate erroneously p_h to be zero or very small. An additional reason for this can be when X_n^j is a multi-valued variable, since in this case it is possible that the sampling algorithm in Figure 1 may miss certain values of X_n^j . To avoid this problem, we apply discrete kernel methods to smooth p_h [8]. Suppose that the hidden variable X_n^j has K values where the probability for each value $h = 1, \dots, K$, is given from (2). Then, the function

$$z = \frac{N}{K-1} \sum_{h=1}^K \frac{(p_h - 1/K)^2}{1/K}$$

denotes the χ^2 Pearson test for the hypothesis that all the categories are equiprobable, standardised by the degrees of freedom $K-1$. A method of smoothing p_h is via the kernel method which gives

$$\hat{p}_h = \sum_{\ell=1}^K p_\ell W_\ell(h, \lambda) \quad (3)$$

where

$$W_\ell(h, \lambda) = \begin{cases} \lambda & \text{if } \ell = h \\ (1-\lambda)/(K-1) & \text{if } \ell \neq h \end{cases}$$

and $\lambda = (N + \alpha)/(N + \alpha K)$ with $\alpha = \begin{cases} z^{-1} & z \geq 1 \\ 1 & z < 1 \end{cases}$. The smoothing parameter α plays the role of placing some mass in value h that may have probability zero. An alternative formulation of (3) is a convex combination of p_h and the uniform estimate $1/K$, that is

$$\hat{p}_h = (1 - \epsilon)p_h + \epsilon/K \quad (4)$$

where $\epsilon = \alpha K/(N + \alpha K)$. The magnitude of α determines a trade-off between bias and variance, since a smaller value of α leads to a less biased, but with higher variance, smoothed estimator, while a larger value of α leads to a smaller variance, but biased, smoothed estimator [8].

To create a sample that will be propagated to the next time step, we need to focus on the hidden variables at time step n that belong to the forward interface \mathbf{FI}_n . That is because every variable $X_n^j \in \mathbf{FI}_n$ belongs to at least one set of parents $\pi(V_{n+1}^j)$ of a variable V_{n+1}^j , and hence a value x_n^j needs to be assigned to it in the sampling algorithm in Figure 1. This can be done easily by generating a value $x_n^{j(i')}$ for sample i' from the established smoothed distribution of X_n^j denoted as $\hat{p}(X_n^j \mid \mathbf{y}_{1:n})$. As a result, the SPF algorithm for monitoring in DBNs consists of two consecutive steps: sampling and smoothing. Schematically, for $i = 1, \dots, N$, the SPF works according to

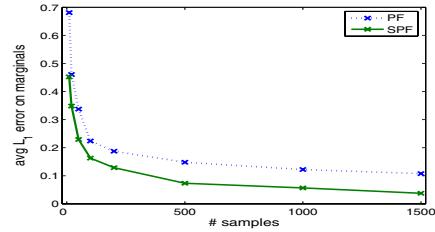


Figure 2: Error as a function of samples.

$$\{\mathbf{x}_n^{(i)}, \tilde{\omega}_n^{(i)}\} \longrightarrow \{x_n^{j(i')}, X_n^j \in \mathbf{FI}_n, \hat{p}(X_n^j \mid \mathbf{y}_{1:n})\} \longrightarrow \{\mathbf{x}_{n+1}^{(i)}, \tilde{\omega}_{n+1}^{(i)}\}$$

The samples thus generated by SPF will not suffer from the impoverishment of the support of the sample that is inevitable with PF. We must also ensure that these samples converge to random samples generated by the true distribution $p(\mathbf{x}_n \mid \mathbf{y}_{1:n})$ as N goes to infinity. Intuitively, this will happen if SPF resembles PF as N becomes larger. From (4) we notice that $\hat{p}_h \rightarrow p_h$ as $N \rightarrow \infty$, and thus we are certain that the estimate given by SPF converges to the true one.

To study the performance of the SPF we performed experiments on the *Mildew* model [3]. In doing so, we randomly created a transition and an observation model for the network where we assume that every variable could take 4 values, and subsequently generated an observation sequence. Our goal was to compare the results at each time step given by the PF and the SPF with the correct distributions computed using exact inference, which is feasible for a model of this size. We used the L_1 -norm to compute the average error on the marginal probability distributions of all the hidden variables. Figure 2 shows the error as a function of the number of samples, where we report the average error over the entire run.

4 Conclusions

We have proposed a sequential importance sampling algorithm, called SPF, to perform inference in DBNs with discrete state space. Our algorithm extends the standard PF in a sense that it replaces the resampling step with a smoothing step. We showed that the smoothing step can be done efficiently using discrete kernel methods that have the effect of placing some mass in hidden states that have erroneously been estimated to have probability zero.

REFERENCES

- [1] A. Doucet. On sequential simulation-based methods for Bayesian filtering. *Technical report CUED/F-INFENG/TR 310*, Department of Engineering, Cambridge University, 1998.
- [2] N.J. Gordon, D.J. Salmond and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F*, 140(2): 107–113, 1993.
- [3] U. Kjaerulff. dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11: 89–111, 1995.
- [4] D. Koller and U. Lerner. Sampling in factored dynamic systems. In A. Doucet, N. De Freitas and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, New York, 2001.
- [5] J.S. Liu and R. Chen. Sequential Monte Carlo methods for dynamical systems. *Journal of the American Statistical Association*, 93: 1032–1044, 1998.
- [6] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. diss, University of California Berkley, 2002.
- [7] C. Musso, N. Oudjane and F. Le Gland. Improving regularised particle filters. In A. Doucet, N. De Freitas and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, New York, 2001.
- [8] J.S. Simonoff. Smoothing categorical data. *Journal of Statistical Planning and Inference*, 47: 41–69, 1995.

Goal Revision for a Rational Agent

Célia da Costa Pereira¹ and Andrea G.B. Tettamanzi¹ and Leila Amgoud²

Abstract. We propose a general framework to represent changes in the mental state of a rational agent due to the acquisition of new information and/or to the arising of new desires; fundamental postulates and properties of the function which generates the goal set are also provided.

1 Introduction

While constructing plans is central in planning, the problem of how goals arise also needs to be addressed. That problem may be approached as in over-subscription planning [8], or as a revision problem, in a spirit akin to belief revision.

Although there has been much discussion on belief revision [6, 5, 10], goal revision has not received much attention. The few works on goal change [4, 9, 7] do not build on belief revision. Their main lack is that agents do not use their own knowledge for revising goals.

We propose an approach for dynamically constructing the goal set to be pursued by a rational agent, by considering changes in its mental state. The key to understanding goal change lies in the fundamental assumption, which is a consequence of agent rationality, that the set of desires an agent pursues must only depend on the agent's state, and not on the history of its previous states.

2 Preliminaries

Following [3], we distinguish two disjoint sets of formulas: the set \mathcal{D} of all possible desires of agents and the set \mathcal{K} of all possible knowledge items.

Definition 1 (Desire-generating Rule) A desire-generating rule is an expression of the form $b_1 \wedge \dots \wedge b_n \wedge d_1 \wedge \dots \wedge d_m \Rightarrow d$, where $b_i \in \mathcal{K}$ and $d_j, d \in \mathcal{D}$, with $d \neq d_j$ for all j .

Definition 2 (Planning Rule) A planning rule is an expression of the form $d_1 \wedge \dots \wedge d_n \rightarrow d$, where $d_i, d \in \mathcal{D}$, $d \neq d_i$ for all i .

We shall denote $\text{lhs}(R)$ the set of literals that make up the conjunction on the left-hand side of a rule R , and $\text{rhs}(R)$ the literal on the right-hand side of R . Given a set S of rules, $\text{rhs}(S) = \{\text{rhs}(R) : R \in S\}$.

Definition 3 (Agent's bases) An agent is equipped with four bases:

- belief base: $\mathcal{B} \subseteq \mathcal{K}$;
- desire base: $\mathcal{J} \subseteq \mathcal{D}$;
- desire-generating rule base: \mathcal{R}_J ;
- planning rules base: \mathcal{P} .

¹ University of Milan, Italy, email: pereira,tettamanzi@dti.unimi.it

² IRIT-CNRS, France, email: amgoud@irit.fr

The state of an agent is completely described by a 4-tuple $\mathcal{S} = \langle \mathcal{B}, \mathcal{R}_J, \mathcal{J}, \mathcal{P} \rangle$. For the sake of simplicity, we assume here that the planning rules are given and fixed.

Definition 4 (Active Desire-generating Rule) A desire-generating rule R is active in \mathcal{S} iff $\mathcal{S} \models \text{lhs}(R)$.

The activation of a desire-generating rule brings about changes in the desire base of an agent. These changes, in turn, may cause the activation/deactivation of other rules. Let $\text{Act}_z^{\mathcal{S}}$ be the set of all desire-generating rules activated by belief or desire z in \mathcal{S} , and $\text{Deact}_z^{\mathcal{S}}$ the set of all desire-generating rules deactivated by belief or desire z in \mathcal{S} .

We do not expect a rational agent to formulate desires out of whim, but based on some rational argument. To model that state of affairs, desire-generating rules play the role of rational arguments and we define a desire to be justified as follows.

Definition 5 (Justified Desire) A desire d is justified in state \mathcal{S} iff d is in \mathcal{J} , i.e., there is an active desire-generating rule R in \mathcal{R}_J such that $\text{rhs}(R) = d$.

3 Changes in the State of an Agent

The state \mathcal{S} of an agent may change either because of the acquisition of a new belief b , in which case the well known AGM operator $*$ for belief revision [1] is applied to \mathcal{B} , or because a new desire d arises. A desire d is retracted from the desire set \mathcal{J} if and only if d becomes not justified, i.e., all active desire-generating rule such that $\text{rhs}(R) = d$ become inactive. A desire d is added to a desire set \mathcal{J} if and only if the new information activates a desire-generating rule R with $\text{rhs}(R) = d$.

If $A_b^{\mathcal{S}}$ be the set of desires acquired because of the new belief b , $A_b^{\mathcal{S}} = \text{rhs}(\text{Act}_b^{\mathcal{S}})$; if $L_b^{\mathcal{S}}$ is the set of desires lost because of the acquisition of the new belief b , $L_b^{\mathcal{S}} = \{d : d \in \text{rhs}(\text{Deact}_b^{\mathcal{S}}) \wedge \neg \exists R (\mathcal{S} \models \text{lhs}(R) \wedge R \notin \text{Deact}_b^{\mathcal{S}} \wedge \text{rhs}(R) = d)\}$.

Let us denote by \oplus a desire revision operator. Then, $\mathcal{J} \oplus b = (\mathcal{J} \cup A_b^{\mathcal{S}}) \setminus L_b^{\mathcal{S}}$. It is easy to verify that $A_b^{\mathcal{S}} \cap L_b^{\mathcal{S}} = \emptyset$, for all state \mathcal{S} .

Proposition 1 The order in which two different pieces of information b_i and b_j are acquired does not affect the content of the final set of desires: $(\mathcal{J} \oplus b_i) \oplus b_j = (\mathcal{J} \oplus b_j) \oplus b_i$.

For the sake of simplicity, we consider that a new desire arises when a desire-generating rule with an empty left hand side is inserted into \mathcal{R}_J . Let us denote by \otimes the operator for updating a desire-generating rule base. When a new desire d arises, the desire-generating rule $\Rightarrow d$ is added to \mathcal{R}_J , d is added to \mathcal{J} , all desire-generating rules R in $\text{Act}_d^{\mathcal{S}}$ become activated, and all desires in the right hand side of these rules are also added to \mathcal{J} . Therefore, $\mathcal{J} \oplus d = \mathcal{J} \cup \{d\} \cup \{\text{rhs}(R), \forall R \in \text{Act}_d^{\mathcal{S}}\}$, and $\mathcal{R}_J \otimes d = \mathcal{R}_J \cup \{\Rightarrow d\}$.

Proposition 2 *The order in which two different desires d_1 and d_2 arise does not affect the final state \mathcal{S} of an agent: $\mathcal{J} \oplus d_1 \oplus d_2 = \mathcal{J} \oplus d_2 \oplus d_1$, and $\mathcal{R}_J \otimes d_1 \otimes d_2 = \mathcal{R}_J \otimes d_2 \otimes d_1$.*

Proposition 3 *Let b be a new piece of information and let d be a new desire for an agent. The new set of desires obtaining from the old set of desires \mathcal{J} by considering b and d does not depend of the order in which these two news components are known by the agent: $\mathcal{J} \oplus b \oplus d = \mathcal{J} \oplus d \oplus b$, and $\mathcal{R}_J \otimes b \otimes d = \mathcal{R}_J \otimes d \otimes b$.*

4 Comparing Desires and Sets of Desires

An agent may have many desires. However, it is essential to be able to represent the fact that not all desires have the same importance or urgency for a rational agent. One approach would be to define a function $u : \mathcal{D} \rightarrow \mathbb{R}$ which associates a real value, utility, to all desires. An alternative approach would be to establish a (partial or total) ordering among desires. In either case, we can define a reflexive and transitive preference relation \succeq between desires, by saying that $d \succeq d'$ iff the agent desires d at least as much as it desires d' . If utilities are defined, $d \succeq d'$ iff $u(d) \geq u(d')$. The \succeq relation can be easily extended from desires to sets of desires: we omit the details due to lack of space.

5 Revising Goal Sets

The main point about desires is that we expect a rational agent to try and manipulate its surrounding environment to fulfill them. In general, not all desires can be fulfilled at the same time, especially when they are conflicting [2]. Therefore, a rational agent will select a justified, coherent and feasible set of desires to realize.

Definition 6 (Coherence of a Desire) *A desire d is coherent, w.r.t. \mathcal{S} , with a set of desires \mathcal{J} iff $\exists \mathcal{I} \models \mathcal{S}$, s.t. $\forall w \in \mathcal{J}, \mathcal{I} \models d \wedge w$.*

Definition 7 (Partial plan) *A partial plan for achieving desire d is a pair $\langle H, d \rangle$ such that: $d \in \mathcal{J}$ and $H = \{d_1, \dots, d_n\}$ if there exists a planning rule $P = d_1 \wedge \dots \wedge d_n \rightarrow d$.*

Definition 8 (Complete Plan) *A complete plan [2] for achieving a desire d is a pair $\langle C, d \rangle$ with $d \in \mathcal{J}$ and C is a finite tree such that its root is a partial plan $\langle H, d \rangle$, a node $\langle \{d_1, \dots, d_n\}, d' \rangle$ has exactly n children $\langle H_1, d_1 \rangle, \dots, \langle H_n, d_n \rangle$, where $\langle H_i, d_i \rangle$ is a partial plan for d_i , and the leaves are partial plans of the form $\langle \emptyset, d_l \rangle$. The function $\text{Des}(C)$ returns the set of all desires in C .*

Definition 9 (Feasible Desire) *A desire d is feasible if there exists a complete plan $\langle C, d \rangle$ to achieve it.*

Definition 10 (Conflict between complete plans) *Two complete plans $\langle C, d \rangle$ and $\langle C', d' \rangle$ are conflicting in a state \mathcal{S} , denoted as $\langle C, d \rangle \bowtie_{\mathcal{S}} \langle C', d' \rangle$, iff $\forall \mathcal{I} \models \mathcal{S}, \mathcal{I} \not\models \text{Des}(C) \cup \text{Des}(C') \cup \mathcal{B}$.*

Definition 11 (Feasible set of Desires) *A set of feasible desires $\{d_1, \dots, d_n\}$ with the respective complete plans $\langle C_1, d_1 \rangle, \dots, \langle C_n, d_n \rangle$, is said to be feasible in state \mathcal{S} iff the plans C_i are not conflicting, i.e., iff there is an interpretation \mathcal{I} such that $\mathcal{I} \models \mathcal{S}$ and $\mathcal{I} \models \text{Des}(C_1) \cup \dots \cup \text{Des}(C_n) \cup \mathcal{B}$.*

Definition 12 (Consistency of a desire with a set) *A feasible desire d is consistent, w.r.t \mathcal{S} , with a set \mathcal{J} of desires, $\text{cons}(d|\mathcal{J})$, iff d is coherent with \mathcal{J} , justified and $\{d\} \cup \mathcal{J}$ is a feasible set.*

Definition 13 (Goal Set) *A goal set is a set of desires $\mathcal{G} \subseteq \mathcal{J}$ such that, $\forall d \in \mathcal{G}, \text{cons}(d|\mathcal{G})$.*

5.1 Postulates for Goal Revision

In general, given a set of desires \mathcal{J} , there are many possible goal sets $\mathcal{G} \subseteq \mathcal{J}$. However, a rational agent in state \mathcal{S} will elect as the set of goals it is pursuing one precise goal set \mathcal{G}^* , which depends on \mathcal{S} .

Let us call G the function which maps a state \mathcal{S} into the goal set elected by a rational agent in state \mathcal{S} : $\mathcal{G}^* = G(\mathcal{S})$. Let \odot be the goal revision operator. The goal election function G must obey three fundamental postulates:

- (**G ⊙ 1**) $\forall \mathcal{S}, G(\mathcal{S})$ is a goal set;
- (**G ⊙ 2**) $\forall \mathcal{S}$, if $\mathcal{G} \subseteq \mathcal{J}$ is a goal set, then $G(\mathcal{S}) \succeq \mathcal{G}$, i.e., a rational agent always selects the most preferable goal set.

A new desire d may become a goal in state $\mathcal{S}_d = \langle \mathcal{B}, \mathcal{J} \oplus d, \mathcal{R}_J \otimes d, \mathcal{P} \rangle$ either if d is consistent with $G(\mathcal{S})$ or if d is not consistent with $G(\mathcal{S})$, but it is preferred to all goals whose complete plans are all conflicting with all its complete plans. Therefore, revising the goal set $G(\mathcal{S})$ must be equivalent to updating the state of the agent, in particular \mathcal{J} and \mathcal{R}_J , then electing the new goal set by means of function G .

Proposition 4 $G(\mathcal{S}) \odot d = G(\mathcal{S}_d)$.

6 Conclusion

Goal revision is more complex than AGM belief revision, because the last desire does not always prevail on previous desires, and a completely different approach is required for a satisfactory treatment.

Many simplifying hypotheses have been made, but we plan on relaxing them in future work.

REFERENCES

- [1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson, ‘On the logic of theory change: Partial meet contraction and revision functions’, *J. Symb. Log.*, **50**(2), 510–530, (1985).
- [2] L. Amgoud, ‘A formal framework for handling conflicting desires’, in *Proc. of the ECSQARU-2003 Conference*, pp. 552–563, (2003).
- [3] L. Amgoud and I. Rahwan, ‘An argumentation-based approach for practical reasoning’, in *To Appear in Proc. of the AAMAS-2006 Conference*, (2006).
- [4] John Bell and Zhisheng Huang, ‘Dynamic goal hierarchies’, in *PRI-CAI ’96: Proceedings from the Workshop on Intelligent Agent Systems, Theoretical and Practical Issues*, pp. 88–103, London, UK, (1997). Springer-Verlag.
- [5] Jon Doyle, ‘Rational belief revision’, in *KR’91: Principles of Knowledge Representation and Reasoning*, eds., James F. Allen, Richard Fikes, and Erik Sandewall, pp. 163–174, San Mateo, California, (1991). Morgan Kaufmann.
- [6] P. Gärdenfors, ‘The dynamics of belief systems: Foundations vs. coherence’, *Revue Internationale de Philosophie*, (1989).
- [7] Steven Shapiro, Yves Lespérance, and Hector J. Levesque, ‘Goal change’, in *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 582–588, Edinburgh, Scotland, UK, (July 30–August 5 2005). Professional Book Center.
- [8] David E. Smith, ‘Choosing objectives in over-subscription planning’, in *Proceedings of ICAPS 2004*, eds., S. Zilberstein, J. Koehler, and S. Koenig, pp. 393–401 (June 3–7 2004). AAAI.
- [9] John Thangarajah, Lin Padgham, and James Harland, ‘Representation and reasoning for goals in bdi agents’, in *CRPITS ’02: Proceedings of the twenty-fifth Australasian conference on Computer science*, pp. 259–265, Darlinghurst, Australia, Australia, (2002). Australian Computer Society, Inc.
- [10] M-A. Williams, ‘Towards a practical approach to belief revision: Reason-based change’, in *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, eds., L.C. Aiello and C. Shapiro, pp. 412–421. Morgan Kaufmann Publishers, (1996).

A Redundancy-based Method for Relation Instantiation from the Web

Viktor de Boer and Maarten van Someren and Bob J. Wielinga ¹

Abstract. The Semantic Web requires automatic ontology population methods. We developed an approach, that given existing ontologies, extracts instances of ontology relations, a specific subtask of ontology population. We use generic, domain independent techniques to extract candidate relation instances from the Web and exploit the redundancy of information on the Web to compensate for loss of precision caused by the use of these generic methods. The candidate relation instances are then ranked based on co-occurrence with a seed set. In an experiment, we extracted instances of the relation between artists and art styles. The results were manually evaluated against selected art resources.

1 INTRODUCTION

The ongoing project of the Semantic Web calls for (semi-)automatic methods for the construction of ontologies (ontology learning) and knowledge bases (ontology population)[3]. In this paper, we describe a method for *relation instantiation*, a subtask of ontology population.

We define a (partly) populated ontology as a set of labeled classes (the domain concepts) C_1, \dots, C_n , hierarchically ordered by a subclass relation. Non-hierarchical relations between concepts are also defined ($R : C_i \times C_j$). We also have a knowledge base containing instances of the ontology concepts. The task of relation instantiation is to identify for a single instance i of C_i for which instances j of C_j , the relation $R(i,j)$ is true given the information in the corpus. In this paper we assume that R is not a one-to-one relation (The instance i is related to multiple instances of C_j). We also assume that we know all instances of C_j and have a method available that recognizes these elements in the documents in our corpus. For a textual corpus such as the Web, this implies that the instances must have a textual label.

2 THE REDUNDANCY METHOD

Current approaches for Information Extraction or Question Answering tasks could also be used for ontology population. However, the performances of methods such as [2] depend on the specific structure or domain of the corpus. We designed our method to be structure- and domain-independent. Also, methods that use some form of supervised Machine Learning assume a large number of tagged example instances to be

able to learn patterns for extracting new instances and this is a serious limitation for large scale use. Our method requires only a small amount of examples that are used as a seed set.

Our approach incorporates generic methods that do not rely on assumptions about the domain or the type of documents in the corpus. By using these general methods for the extraction, we will lose in precision since the general methods are not optimized for a specific corpus or domain. However, since we use more generic methods, we are able to extract information from a greater number of sources. The main assumption behind our method is that because of the redundancy of information on the Web and because we are able to combine information from heterogeneous sources, we can compensate for this loss of precision.

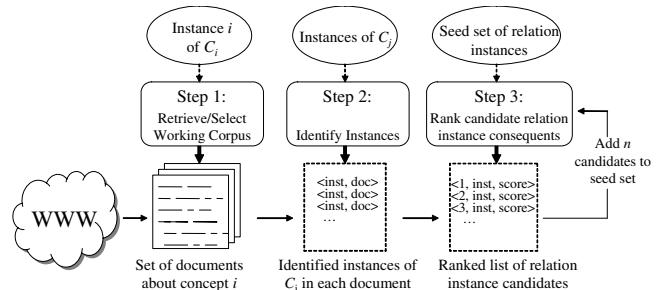


Figure 1. Outline of the method

The method consists of three steps, shown in Figure 1. We first construct a 'working corpus' by feeding the label(s) of the instance i to the Google search engine. The size of this working corpus is a parameter of the method.

In step 2, we identify the instances of the concept C_j in the documents of the working corpus. For this, we use a Named Entity Recognition module and match the results to the instances of C_j in our populated ontology, this yields our candidate relation instances.

In step 3, the method combines the evidence from the different documents to produce a ranking for these candidates. We base this ranking on the assumption that on average in individual web pages, a target relation is either well represented (the web page contains a number of correct right-hand side instances) or not represented (it contains few or none of these instances).

We therefore calculate a Document Score DS for each document. This is the probability that for all candidates in that document the relation R holds, according to the seed set. This is equal to the number of identified instances that are in the

¹ Human-Computer Studies Laboratory, Informatics Institute, Universiteit van Amsterdam, email: {vdeboer,maarten,wielinga}@science.uva.nl

seed set divided by the total number of candidate instances in that document. We then combine all evidence for each of the candidate instances by taking the average of DS over all used documents in the corpus resulting in an Instance Score IS for each candidate instance.

We then add the candidate with the highest value of IS to the seed set and iterate by recalculating all DS and IS , based on the expanded seed set. The method iterates up to a threshold on the number of iterations or a drop in the Instance Scores. In Section 3, we explore the effects of these thresholds.

3 EXTRACTING ART STYLE-ARTISTS RELATION

We tested our method in the cultural heritage domain. We used two well-known art thesauri as our partly populated ontologies: the AAT[4] and the ULAN[5]. In this experiment we extracted the instances of the relation 'has_artist' between `aat:Art Style` and `ulan:Artist`. We tested the method for nine art styles².

We first populated the seed set with three well-known artists associated with that art style. Then in Step 1, 1000 pages were extracted as a working corpus by querying Google with the labels of the art style instances. In Step 2, we used the Person Name Extractor from the tOKO toolkit[1] and matched the results to the ULAN. In Step 3, the DS and IS scores were calculated and for each art style. We evaluated the results of 40 iterations by having two annotators manually score each of the 40 retrieved relation instances as 'correct' or 'incorrect' for each art style. The annotators were allowed to consult a fixed set of encyclopedic web sources. Inter-annotator agreement (Cohen's Kappa) = 0.83. Finally, consensus was reached to calculate precision.

We first illustrate the results for a single art style: 'Neue Sachlichkeit' ('New Objectivity'). Figure 2 shows the IS for the top artists and value of precision for all 40 iterations.

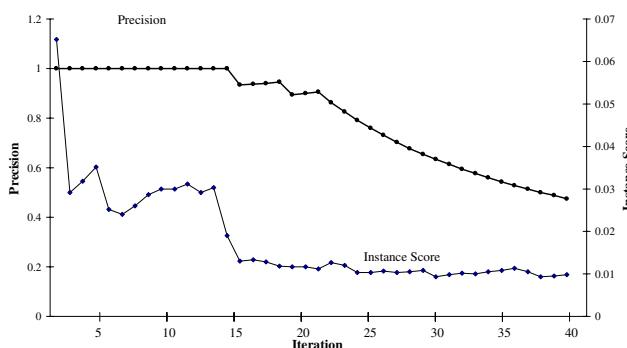


Figure 2. IS and precision for 'Neue Sachlichkeit'

The drop in precision co-occurs with a drop in IS . We can use this drop in IS as a threshold. We stop adding relation instances to the knowledge base if the value of IS of the next candidate instance is less than some drop factor, DF , multiplied by the maximum of the Instance Scores up to that iteration. We also stop adding instances after an absolute maximum number of iterations has been reached (Max). In the above example, setting DF to 0.2 and Max to 40, leads to a precision of 0.933, with 15 correct relations added.

² Art Deco, Art Nouveau, Cubism, Dada, Expressionism, Impressionism, Neo-Impressionism, Neue Sachlichkeit and Surrealism

Table 1. Average precision (prec) and total number of correct extractions (ex) for the nine Art Styles

DF	Max							
	10		20		30		40	
prec	ex	prec	ex	prec	ex	prec	ex	
0	0.856	77	0.806	145	0.722	195	0.650	234
0.1	0.856	77	0.806	145	0.721	193	0.648	228
0.2	0.856	77	0.799	137	0.776	179	0.746	197
0.3	0.865	73	0.842	117	0.830	138	0.810	144
0.4	0.857	62	0.834	96	0.826	114	0.824	120
0.5	0.902	55	0.878	86	0.868	103	0.866	109
0.6	0.924	46	0.896	67	0.882	81	0.880	87

In Table 1, we list both the average precision and the total sum of the number of correct relation instances extracted for the nine art styles for 24 combinations of the two threshold parameters DF and Max . The lowest value for precision is 0.65. This occurs at $DF=0$ (the drop in the Instance Score is not used to set the threshold) and $Max=40$. In that case, for the nine art styles, all 360 (9×40) extractions are added to the knowledge base, of which 234 are evaluated correct. The highest precision, 0.924, is reached at $DF=0.6$ and $Max=10$, with only 46 correct relation instances added to the knowledge base. We observe relatively high values for precision and a tradeoff between precision and number of correct extractions comparable to that of the traditional precision/recall tradeoff.

4 CONCLUSIONS

Considering the method uses very generic methods and intuitive ranking scores, the results are encouraging but also suggest that further processing of the results could improve the relation instantiation. Analysis of the working corpus showed the documents were highly heterogeneous in structure and language. How much redundancy helped is a topic for further research. Improvement in the Person Name Extraction module or combining different Person Name Extractors could improve the extraction. Also, other measures for DS and IS could be considered. An obvious direction for further research is to test this method on other relations in other domains.

ACKNOWLEDGEMENTS

This research was supported by the MultimediaN project (www.multimedian.nl) funded through the BSIK programme of the Dutch Government.

REFERENCES

- [1] A. Anjewierden, B.J. Wielinga, and R. de Hoog, 'Task and domain ontologies for knowledge mapping in operational processes', *Metis Deliverable 4.2/2003, University of Amsterdam*, (2004).
- [2] N. Kushmerick, D. Weld, and R. Doorenbos, 'Wrapper induction for information extraction', in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, p. 729737, (1997).
- [3] A. Maedche and S. Staab, 'Ontology learning for the semantic web', *IEEE Intelligent Systems*, **13**, 993, (2001).
- [4] The Getty Foundation, 'Aat: Art and architecture thesaurus', <http://www.getty.edu/research/tools/vocabulary/aat/>.
- [5] The Getty Foundation, 'Ulan: Union list of artist names', <http://www.getty.edu/research/tools/vocabulary/ulan/>.

Norms with deadlines in Dynamic Deontic Logic

Demolombe Robert¹ and Bretier Philippe² and Louis Vincent³

Abstract.

In this paper we extend the logical framework defined by K. Segerberg about norms and actions to norms that refer to deadlines. We also characterize the circumstances where these norms are violated.

1 Introduction

Interactions between artificial agents who are partially autonomous, or between human agents, are restricted by regulations that define the obligations, forbiddances or permissions they have to fulfil.

There are many papers in the literature about the formalization of regulations, or norms in general. Some norms are about the obligation to do an action. They combine obligations and actions, and there are several proposals for their formalization in the framework of modal logics [9, 10, 8, 5]. However, as far as we know, Dignum et al., in [7, 6], and Broersen et al., in [2, 3], are the only researchers who have taken into account a fundamental feature of obligations to do, which is the notion of **deadline**.

For example, if we just say that it is obligatory for some agent to pay some bill, without specifying any deadline, then the obligation will never be violated. Even if the customer never pays!

This example is clear enough to understand that the deadline is a constitutive component of an obligation to do. The objective of this paper is to present a new definition of obligations to do with deadlines.

2 Segerberg's dynamic deontic logic

Obligations to do raise a particular formalization problem with regard to obligations to be, because formulae in the scope of an obligation must denote a proposition, and actions are denoted by terms (see [9]).

The solution proposed by Segerberg in [10, 9] is to interpret an obligation to do an action as an obligation to have done this action.

His logic is defined by its semantics. The primitive notion is the notion of **point** (also called “**instant**”), which is similar to a possible world in a Kripke model. A **path** is a sequence of points that can be understood as a given evolution of the world.

An **event type** (an *event* for short) is a finite set of paths. Each path can be viewed as a particular realization of the event type.

An **individual action** is a tuple $\langle i, e, p \rangle$, where i denotes an agent, e an event type and p a path. Intuitively, i is the agent who realized the event type e along the path p .

¹ ONERA Toulouse, France, email: Robert.Demolmbe@cert.fr

² France Telecom Research and Development, France, email: philippe.bretier@francetelecom.com

³ France Telecom Research and Development, France, email: vincent.louis@francetelecom.com

A **history** is a sequence of individual actions, which can be finite or infinite, of the form: $\langle i_0, e_0, p_0 \rangle \langle i_1, e_1, p_1 \rangle \dots \langle i_n, e_n, p_n \rangle$

We say that “ i does e in h ”, if p is some path in e and $\langle i, e, p \rangle$ is an individual action of the history h . That is, in formal terms: $\exists h', h''$ such that $h = h' \langle i, e, p \rangle h''$ (h' and h'' may be empty).

We use the following notations:

- ef is the event consisting of e immediately followed by f ,
- pq is the path made up by p immediately followed by q .

We say that the histories h and h' are equivalent, and that is denoted by $h \approx h'$, iff $\exists g, g'^4$

$$h = g \langle i, e_0, p_0 \rangle \dots \langle i, e_{n-1}, p_{n-1} \rangle g' \text{ and} \\ h' = g \langle i, e_0, \dots, e_{n-1}, p_0, \dots, p_{n-1} \rangle g'$$

It is assumed that for any given history there is a definite set of possible continuations. The set of complete continuations of h is denoted by: $cont(h)$. This set represents the set of possible futures when we are at the last point of h .

It is also assumed that for every h the set $cont(h)$ can be partitioned into two categories: the set of continuations that conform to the Norm (also called “ideal histories”), denoted by $norm(h)$, and those that do not.

The formal language is the language of a propositional multi modal logic [4] with the following additional modal operators:

$[H]\phi$: it is historically necessary that ϕ .

$[D]\phi$: it is deontically necessary that ϕ .

$[F]\phi$: it will always be the case that ϕ .

$[P]\phi$: it always was the case that ϕ .

The satisfiability conditions are defined using the relation:

$$(h, g) \models \phi$$

whose intuitive meaning is: ϕ is true at a point which is the last point of the past history h , and the first point of the future history g .

If ϕ is an atomic formula A , we have $(h, g) \models A$ iff the last point of h is in the set of points that interprets A . The satisfiability conditions for logical connectives are defined as usual. For the modal operators we have:

$$(h, g) \models [H]\phi \text{ iff } \forall g' \in cont(h)((h, g') \models \phi)$$

$$(h, g) \models [D]\phi \text{ iff } \forall g' \in norm(h)((h, g') \models \phi)$$

$$(h, g) \models [F]\phi \text{ iff } \forall g_0, g_1(g = g_0 g_1 \Rightarrow (h g_0, g_1 \models \phi))$$

$$(h, g) \models [P]\phi \text{ iff } \forall h_0, h_1(h = h_0 h_1 \Rightarrow (h_0, h_1 g) \models \phi)$$

Actions are represented by terms built up with atomic actions and the constructors of sequence and non deterministic choice.

$|\alpha|$ is used to denote an event type which is the interpretation of the action α .

For each agent i we have the event-to-proposition operator:

$done_i(\alpha) : i$ has just finished doing α .

Its formal semantics is defined by:

$$(h, g) \models done_i(\alpha) \text{ iff } \exists h', e, p(p \in e \wedge e = |\alpha| \wedge h \approx h' \langle i, e, p \rangle)$$

⁴ $\exists g, g'$ abbreviates $\exists g \exists g'$, and $\forall g, g'$ abbreviates $\forall g \forall g'$.

We say that: “ i does α in h ” iff $\exists h', h'', e, p (p \in e \wedge e = |\alpha| \wedge h \approx h' < i, e, p > h'')$.

Now it is possible to introduce the two operators that represent obligation or forbiddance to do an action:

$ob_i(\alpha)$: it is obligatory for agent i to have done action α .

$fb_i(\alpha)$: it is forbidden for agent i to have done action α .

Their satisfiability conditions are:

$(h, g) \models ob_i(\alpha)$ iff $\forall g' \in cont^0(h)$

$\neg(i \text{ does } \alpha \text{ in } g') \Rightarrow \forall f \in norm(hg') (i \text{ does } \alpha \text{ in } f)$

$(h, g) \models fb_i(\alpha)$ iff $\forall g' \in cont^0(h)$

$\forall f \in norm(hg') \neg(i \text{ does } \alpha \text{ in } f)$

where $cont^0(h)$ denotes the set of h finite continuations.

The *until* operator is assigned the intuitive meaning:

$(\text{until } \phi)\psi$: ψ holds from the point where we are until ϕ holds

Its formal definition is⁵:

$(h, g) \models (\text{until } \phi)\psi$ iff $\forall g', g'' (g \approx g'g'' \Rightarrow$

$((hg', g'') \models \psi \vee \exists g_0, g_1 (g' \approx g_0g_1 \wedge (hg_0, g_1g'') \models \phi)))$

If there are several points in g where ϕ holds, $(\text{until } \phi)\psi$ guarantees that ψ holds until the point that is just before the first point where ϕ holds.

The operator $(before \phi)\psi$ is assigned the intuitive meaning:

$(before \phi)\psi$: ψ will hold, and it will hold before ϕ holds.

If ϕ and ψ have several occurrences in the future, $(before \phi)\psi$ guarantees that the first occurrence of ψ will hold before the first occurrence of ϕ . The formal definition of *before* is:

$(before \phi)\psi \stackrel{\text{def}}{=} (\text{until } \psi) \neg \phi \wedge \langle F \rangle \psi$

Then, it can be shown that we have the following valid formulas:

$ob_i(\alpha) \leftrightarrow [H](\text{until } done_i(\alpha))[D] \langle F \rangle done_i(\alpha)$

$fb_i(\alpha) \leftrightarrow [H][F][D][F] \neg done_i(\alpha)$

3 Extension to norms with deadlines

Previous definitions are now extended to norms with deadlines. We define the operator $ob_i(\alpha < d)$ whose intuitive meaning is:

$ob_i(\alpha < d)$: it is obligatory for agent i to have done the action α between the present point and the first occurrence of the proposition d . Its formal definition is:

$ob_i(\alpha < d) \stackrel{\text{def}}{=} [H](\text{until } done_i(\alpha) \vee d)[D](before d)done_i(\alpha)$

We have similar operators for forbiddance and permission. Their formal definitions are:

$fb_i(\alpha < d) \stackrel{\text{def}}{=} [H](\text{until } d)[D](\text{until } d) \neg done_i(\alpha)$

$pmi(\alpha < d) \stackrel{\text{def}}{=} [H](\text{until } d) \langle D \rangle (before d)done_i(\alpha)$

In the case where there is no deadline (i.e. d is equivalent to *false*) we get the same definitions as Segerberg.

Violations.

An obligation of the form $ob_i(\alpha < d)$ has been violated at the present instant t , if there is a past instant t_1 where:

- d has occurred, and

- before t_1 there was an instant t_2 where we had $ob_i(\alpha < d)$, and the action α has never been done between t_2 and the first instant after t_2 where d has occurred.

If $ob.viol(i, \alpha, d)$ is used to denote the formula that characterizes the violation of the obligation $ob_i(\alpha < d)$, we have:

$ob.viol(i, \alpha, d) \stackrel{\text{def}}{=} \langle P \rangle (d \wedge \langle P \rangle (ob_i(\alpha < d) \wedge (\text{until } d) \neg done_i(\alpha)))$

A forbiddance of the form $fb_i(\alpha < d)$ has been violated at the present instant t if there is a past instant t_1 where:

- we had $fb_i(\alpha < d)$, and

- between t_1 and the first occurrence of d , α has been done.

If $fb.viol(i, \alpha, d)$ is used to denote the formula that characterizes the violation of the forbiddance $fb_i(\alpha < d)$, we have:

$fb.viol(i, \alpha, d) \stackrel{\text{def}}{=} \langle P \rangle (done_i(\alpha) \wedge \langle P \rangle (fb_i(\alpha < d) \wedge (before d)done_i(\alpha)))$

In the case of permissions there is no violation. Indeed, a permission does not impose to do an action, it just offers the possibility to do an action without any violation of the norms.

4 Conclusion

We have extended the Segerberg's logical framework to norms with deadlines. This general framework proposes solutions to real needs in practical applications.

Other approaches that can be found in the state of the art.

In [8], Horty and Belnap present a logic where histories have a tree structure which has some similarities with Segerberg's structures. The main difference is that, they use a “*stir*” operator where actions are not explicitly represented.

Dignum et al.[7] have defined a temporal logic with tree structures and a *stir* operator. Norms are formalized according to the Anderson's reductionist approach [1], and there is no explicit representation of ideal histories. Moreover deontic modalities cannot be nested with other modal operators.

In Broersen et al. [2], formulas of the form $O_i(\phi, d)$ mean that it is obligatory for agent i that ϕ holds before the deadline d . But there is no explicit reference to the actions, and we see this operator as a representation of obligation to be. In [3], there is no deontic modality and norms are formalized by their violation à la Anderson.

The logical framework that has been presented could be extended in several directions. The first one is to consider obligations to be with deadlines. Another direction could be to have obligations to have started the performance of an action. Finally, an important and difficult pending problem is to find a complete axiomatics for this logic.

REFERENCES

- [1] A. R. Anderson, ‘A reduction of deontic logic to alethic modal logic’, *Mind*, **67**, (1958).
- [2] J. Broersen, M. Dastani, and L. van der Torre, ‘BDIO-CTL: obligations and the specification of agent behaviour’, in *International Joint Conference on Artificial intelligence*, (2003).
- [3] J. Broersen, F. Dignum, V. Dignum, and J.-J. C. Meyer, ‘Designing a deontic logic of deadlines’, in *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science*, eds., A. Lomuscio and D. Nute. Springer, LNAI 3065, (2004).
- [4] B. F. Chellas, *Modal Logic: An introduction*, Cambridge University Press, 1988.
- [5] R. Demolombe and A.J. Jones, ‘Actions and normative positions. A modal-logical approach’, in *Companion to Philosophical Logic*, ed., D. Jacquette, Blackwell, (2002).
- [6] F. Dignum and R. Kuiper, ‘Combining dynamic deontic logic and temporal logic for the specification of deadlines’, in *Thirties HICSS*, (1997).
- [7] V. Dignum, J.-J. Meyer, F. Dignum, and H. Weigand, ‘Formal specification of interaction in agent societies’, in *Second Goddard workshop on Formal Approaches to Agent-Based Systems*, (2002).
- [8] J.F. Horty and N. Belnap, ‘The deliberative STIT: a study of action, omission, ability, and obligation’, *Journal of Philosophical Logic*, **24**, 583–644, (1995).
- [9] K. Segerberg, ‘Some Meinong/Chisholm thesis’, in *Logic, Law, Morality. A festchrift in honor of Lennart Åqvist*, eds., K. Segerberg and K. Sliwinski, volume 51, 67–77, Uppsala Philosophical Studies, (2003).
- [10] K. Segerberg, ‘Intension, Intention’, in *To be announced*, ed., R. Kahle, CSLI Publications, (2004).

⁵ We have slightly modified the Segerberg's definition.

Adaptive Multi-Agent Programming in GTGolog

Alberto Finzi^{1,2} and Thomas Lukasiewicz^{2,1}

Abstract. We present a novel approach to adaptive multi-agent programming, which is based on an integration of the agent programming language GTGolog with adaptive dynamic programming techniques. GTGolog combines explicit agent programming in Golog with game-theoretic multi-agent planning in stochastic games. In GTGolog, the transition probabilities and reward values of the domain must be provided with the model. The adaptive generalization of GTGolog proposed here is directed towards letting the agents themselves explore and adapt these data. We use high-level programs for the generation of both abstract states and optimal policies.

1 INTRODUCTION

We present a novel approach to adaptive multi-agent programming, which is based on an integration of GTGolog [5] with multi-agent reinforcement learning as in [7]. We use high-level programs for both generating abstract states and learning policies over these abstract states. The generation of abstract states exploits the structured representation of the domain in a basic action theory in the situation calculus (SC) [9], along with the high-level control knowledge in a Golog program [9]. A learning process then incrementally adapts the model to the executive context and instantiates the partially specified behavior. To our knowledge, this is the first adaptive approach to Golog interpreting. Differently from classical Golog [9, 3], here the interpreter generates not only complex sequences of actions, but also an abstract state space for each machine state. Similarly to [2, 6], we rely on the SC machinery for state abstraction, but in our system the state generation depends on the program structure. Here, we can take advantage from the deep integration between the SC action theory and Golog programs. Similarly to hierarchical reinforcement learning [4, 1], the choice points of partially specified hierarchical programs are instantiated through reinforcement learning and dynamic programming constrained by the program structure.

2 ADAPTIVE GTGOLOG

In this section, we present AGTgolog for $n \geq 2$ agents. To introduce our framework, we refer to the following scenario inspired by [8].

Example 2.1 We have a game field which is a grid of 9×9 positions, and two agents, a and o , can move one step in the N, S, E, W directions, or remain stationary. Each of them can pick up wood or gold when at a forest or goldmine, respectively, and drop these resources at its base. Each action can fail resulting in a stationary move. A reward is received whenever a unit of wood or gold is brought back to the base of an agent. Any carried object drops when two agents collide. The game is zero-sum, hence, after each move an agent receives

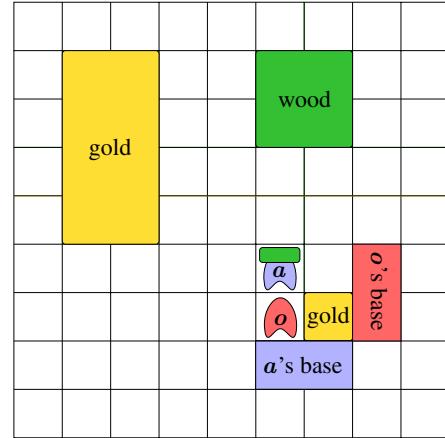


Figure 1. Example Domain

a reward consisting of the reward of its action minus the reward of the adversary action.

Domain Theory of AGTgolog. A *domain theory* of AGTgolog $DT = (AT, ST, OT)$ consists of a basic action theory AT , a *stochastic theory* ST , and an *optimization theory* OT , defined in the SC.

We assume two zero-sum competing agents a and o (called *agent* resp. *opponent*, where the former is under our control, while the latter is not). A *two-player action* is either an action a for agent a , or an action b for agent o , or two parallel actions $a||b$, one for each agent. E.g., $move(a, N)$, $move(o, W)$, and $move(a, N)||move(o, W)$ are two-player actions. The basic action theory AT represents a deterministic dynamic domain through fluent predicates, e.g., $at(q, x, y, s)$ and $holds(\alpha, o, s)$, which are defined by an FOL theory for the initial situation S_0 and *successor state axioms*, e.g.,

$$\begin{aligned} holds(\alpha, o, do(a, s)) \equiv & \quad holds(\alpha, o, s) \wedge \neg dropping(\alpha, a) \wedge \\ & \neg collision(a, s) \vee pickingUp(\alpha, o, a). \end{aligned}$$

It also encodes *action preconditions*, one for each action, e.g.,

$$Poss(pickUpS(\alpha), s) \equiv \neg \exists x (holds(\alpha, x, s)).$$

Given AT , we can specify ST . As usual in the SC [3], stochastic actions are expressed by a finite set of deterministic actions represented by AT . When a stochastic action is executed, then “nature” chooses and executes with a certain probability exactly one of its deterministic actions. E.g., we introduce the stochastic action $moveS(\alpha, m)$ (agent α executes $m \in \{N, S, E, W, stand\}$), associated with the deterministic actions $move(\alpha, m)$ and $move(\alpha, stand)$ representing success and failure, respectively. To encode probabilities in ST (and rewards in OT), we use *state formulas* and *state partitions*. A *state formula* over \vec{x}, s is an SC formula $\phi(\vec{x}, s)$ in which all predicate symbols are fluents, and the only free variables are the non-situation variables \vec{x} and the situation variable s . A *state partition* over \vec{x}, s is a nonempty set of state formulas $P(\vec{x}, s) = \{\phi_i(\vec{x}, s) \mid i \in \{1, \dots, m\}\}$ such that (i) $\forall \vec{x}, s (\phi_i(\vec{x}, s) \Rightarrow \neg \phi_j(\vec{x}, s))$ is valid

¹ Inst. für Informationssysteme, TU Wien, Favoritenstr. 9-11, A-1040 Wien.

² DIS, Università di Roma “La Sapienza”, Via Salaria 113, I-00198 Roma.

for all $i, j \in \{1, \dots, m\}$ with $j > i$, (ii) $\forall \vec{x}, s \bigvee_{i=1}^m \phi_i(\vec{x}, s)$ is valid, and (iii) every $\exists \vec{x}, s \phi_i(\vec{x}, s)$ is satisfiable. For state partitions P_1 and P_2 , we define $P_1 \otimes P_2 = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in P_1, \psi_2 \in P_2, \psi_1 \wedge \psi_2 \neq \perp\}$.

Given a stochastic action a and an associated component n , we specify a state partition $P_{pr}^{a,n}(\vec{x}, s) = \{\phi_j^{a,n}(\vec{x}, s) \mid j \in \{1, \dots, m\}\}$ to group together situations s with common p such that “nature” chooses n in s with probability p , denoted $\text{prob}(a(\vec{x}), n(\vec{x}), s) = p$:

$$\exists p_1, \dots, p_m (\bigwedge_{j=1}^m (\phi_j^{a,n}(\vec{x}, s) \Leftrightarrow \text{prob}(a(\vec{x}), n(\vec{x}), s) = p_j)).$$

In our example, we assume $P_{pr}^{a,n} = \{\top\}$ for each action-component pair, e.g., $\exists p (\text{prob}(\text{pickUpS}(\alpha), \text{pickUp}(\alpha), s) = p)$.

As for the optimization theory OT , for every two-player action a , we specify a state partition $P_{rw}^a(\vec{x}, s) = \{\phi_k^a(\vec{x}, s) \mid k \in \{1, \dots, q\}\}$ to group together situations s with common r such that $a(\vec{x})$ and s assign the reward r to agent a , denoted $\text{reward}(a(\vec{x}), s) = r$:

$$\exists r_1, \dots, r_q (\bigwedge_{k=1}^q (\phi_k^a(\vec{x}, s) \Leftrightarrow \text{reward}(a(\vec{x}), s) = r_k)).$$

In our domain, we define a zero-sum reward function as follows:

$$\begin{aligned} \text{reward}(\alpha, a, s) = r &\equiv \exists r_\alpha (\text{rewAg}(\alpha, a, s) = r_\alpha \wedge \\ &\quad \exists \beta, r_\beta (\text{rewAg}(\beta, a, s) = r_\beta \wedge r = r_\alpha - r_\beta)), \\ \exists r_1, \dots, r_m (\bigwedge_{j=1}^m \phi_j^{\alpha,a}(s) &\Leftrightarrow \text{rewAg}(\alpha, a, s) = r_j). \end{aligned}$$

Here, $\phi_j^{\alpha,a}(\vec{x}, s)$ belongs to $P_{rw}^{\alpha,a}$. Moreover, a utility function associates with every reward v and success probability pr a real-valued utility $\text{utility}(v, pr)$. We assume that $\text{utility}(v, 1) = v$ and $\text{utility}(v, 0) = 0$ for all v . E.g., $\text{utility}(v, pr) = v \cdot pr$.

Syntax of AGTGolog. AGTGolog has the same syntax as standard GTGolog. Given the actions in the domain theory DT , a program p has one of the following forms:

- *Deterministic or stochastic action:* α . Do α .
- *Nondeterministic action choice of a :* $\text{choice}(a : a_1 | \dots | a_n)$. Do an optimal action (for agent a) among a_1, \dots, a_n .
- *Nondeterministic action choice of o :* $\text{choice}(o : o_1 | \dots | o_m)$. Do an optimal action (for agent o) among o_1, \dots, o_m .
- *Nondeterministic joint action choice:* $\text{choice}(a : a_1 | \dots | a_n) \parallel \text{choice}(o : o_1 | \dots | o_m)$. Do any action $a_i \parallel o_j$ with an optimal probability $\pi_{i,j}$.
- *Test action:* $\phi?$. Test the truth of ϕ in the current situation.
- *Action sequence:* $p_1; p_2$. Do p_1 followed by p_2 .
- *Nondeterministic choice of two programs:* $(p_1 \mid p_2)$. Do p_1 or p_2 .
- *Nondeterministic choice of an argument:* $\pi x(p(x))$. Do any $p(x)$.
- *Conditionals, while-loops, and procedures, including recursion.*

E.g., in our example, we define the procedure *tryToPickUp*:

```
proc(tryToPickUp, choice(a: pickUpS(a)|moveS(a, stand))||choice(o: pickUpS(o)|moveS(o, stand))).
```

State Partition Generation. Given an AGTGolog program p , a machine state consists of a subprogram p' of p and a number of steps to go h . A joint state (ϕ, p, h) consists of a state formula ϕ and a machine state (p, h) . Every machine state (p, h) is associated with a state partition, denoted $SF(p, h) = \{\phi_1(\vec{x}, s), \dots, \phi_m(\vec{x}, s)\}$. E.g., for *null program* or *zero horizon* (base case):

$$SF(\text{Nil}, h) = SF(p', 0) = \{\top\}.$$

For *deterministic first program action*, we have:

$$\begin{aligned} SF(a; p', h) = P_{rw}^a(\vec{x}, s) \otimes \{Regr(\phi(\vec{x}, do(a, s)) \wedge Poss(a, s) \mid \\ \phi(\vec{x}, s) \in SF(p', h-1)\} \cup \{\neg Poss(a, s)\} - \{\perp\}. \end{aligned}$$

For $p = \text{tryToPickUp} ; \text{carryToBase}$ and $h = 3$, each program state (e.g., $(p, 3)$ or $(p_1, 2)$, with $p_1 = \text{carryToBase}$) is associated with a suitable state partition, e.g., $SF(p, 3)$ or $SF(p_1, 2)$.

Learning Algorithm. For each joint state σ composed of a machine state (p, h) and a state formula $\phi(\vec{x}, s) \in SF(p, h)$, the learning algorithm generates the best policy $\pi(\sigma)$ for the agent a . The policy is

Algorithm $\text{Learn}(p, h)$

Input: GTGolog program p and finite horizon h .

Output: optimal policy $\pi(\phi, p, h)$ for all $\phi \in SF(p, h)$.

1. $\alpha := 1$;
2. **for each** joint state σ **do** $\langle v, pr \rangle(\sigma) := \langle 1, 1 \rangle$;
3. **repeat**
4. estimate $\phi \in SF(p, h)$;
5. $\text{Update}(\phi, p, h)$;
6. $\alpha := \alpha \cdot \text{decay}$
7. **until** $\alpha < \varepsilon$;
8. **return** $(\pi(\phi, p, h))_{\phi \in SF(p, h)}$.

Figure 2. Algorithm $\text{Learn}(p, h)$

obtained from (p, h) by replacing every single-agent choice in p by a single action, and every multi-agent choice by a collection of probability distributions $\pi(\sigma)$, one over the actions of each agent. We deploy a hierarchical version of Q-learning described in Figure 2.

We initialize the learning rate α (decaying at each cycle according to decay) and the variables $\langle v, pr \rangle(\cdot)$ representing the current value of the v -function in a joint state. At each cycle, the current state $\phi \in SF(p, h)$ is estimated. Then, from the joint state $\sigma = (\phi, (p, h))$, the procedure $\text{Update}(\phi, p, h)$ updates the values $\langle v, pr \rangle(\sigma)$ during an execution of the program p with horizon h . $\text{Update}(\phi, p, h)$ is inductively defined w.r.t. the structure of the program. E.g., if $p = a ; p'$, where a is deterministic and executable in ϕ , then $\text{Update}(\phi, p, h)$ executes a and gets *reward*; upon the $\text{Update}(do(a, \phi), p', h-1)$ execution, we have the updates $\langle v, pr \rangle(\sigma) := \langle (1 - \alpha) \cdot v(\sigma) + \alpha \cdot (\text{reward} + \gamma \cdot v(do(a, \phi), p', h-1)), pr(do(a, \phi), p', h-1) \rangle$ and $\pi(\sigma) := a ; \pi'(do(a, \phi), p', h-1)$. If $p = \text{choice}(a : a_1 | \dots | a_n) \parallel \text{choice}(o : o_1 | \dots | o_m) ; p'$, then the update step is associated with a Nash equilibrium (as in [7]): $(\pi_a, \pi_o) := \text{selectNash}(\{r_{i,j} = \text{utility}(\langle v, pr \rangle(\phi, a : a_i \parallel o : o_j ; p', h)) \mid i, j\})$; where $\langle v, pr \rangle(\sigma) := \sum_{i=1}^n \sum_{j=1}^m \pi_a(a_i) \cdot \pi_o(o_j) \cdot \langle v, pr \rangle(\phi, a : a_i \parallel o : o_j ; p', h)$. E.g., in our domain, we run the learning algorithm to instantiate a policy for the program $p = \text{tryToPickUp} ; \text{carryToBase}$, with horizon 3, i.e., $\text{Learn}(p, 3)$. The agent runs several times p (with horizon 3) playing against the opponent until the learning ends and the variables $\langle v, pr \rangle$ are stabilized for each joint state of the program p .

Acknowledgements. This work was supported by the Austrian Science Fund Project P18146-N04 and by a Heisenberg Professorship of the German Research Foundation (DFG). We thank the reviewers for their constructive comments, which helped to improve this work.

REFERENCES

- [1] D. Andre and S. J. Russell, ‘State abstraction for programmable reinforcement learning agents’, in *Proc. AAAI-2002*, pp. 119–125.
- [2] C. Boutilier, R. Reiter, and B. Price, ‘Symbolic dynamic programming for first-order MDPs’, in *Proc. IJCAI-2001*, pp. 690–700.
- [3] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun, ‘Decision-theoretic, high-level agent programming in the situation calculus’, in *Proc. AAAI-2000*, pp. 355–362.
- [4] T. G. Dietterich, ‘The MAXQ method for hierarchical reinforcement learning’, in *Proc. ML-1998*, pp. 118–126.
- [5] A. Finzi and T. Lukasiewicz, ‘Game-theoretic agent programming in Golog’, in *Proc. ECAI-2004*, pp. 23–27.
- [6] C. Gretton and S. Thiebaut, ‘Exploiting first-order regression in inductive policy selection’, in *Proc. UAI-2004*, pp. 217–225.
- [7] M. L. Littman, ‘Markov games as a framework for multi-agent reinforcement learning’, in *Proc. ICML-1994*, pp. 157–163.
- [8] B. Marthi, S. J. Russell, D. Latham, and C. Guestrin, ‘Concurrent hierarchical reinforcement learning’, in *Proc. IJCAI-2005*, pp. 779–785.
- [9] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, MIT Press, 2001.

Formalizing Complex Task Libraries in Golog

Alfredo Gabaldon¹

Abstract. We present an approach to building libraries of *tasks* in complex action languages, such as Golog, for query answering. Our formalization is based on a situation calculus framework that allows probabilistic, temporal actions. Once a knowledge base is built containing domain knowledge including type information and a library of tasks and the goals they can achieve, we are interested in queries about the achievability of goals. We consider cases where, using domain object type and goal information in the KB, a user is able to get specific answers to a query while leaving some of the details for the system to figure out. In some cases where the specifics are missing from the KB, the user is provided with possible alternative answers that are compatible with the incomplete information in the KB. This approach is being explored in the context of a military operations planning domain for decision support.

1 Introduction

Complex activity planning as required in military operations, the construction industry, software development, etc., is usually carried out by expert hand from scratch or by reuse and adaptation of solutions to similar, previous problems. Computer assistance is used for solving the resulting scheduling problem, usually only after the set of activities necessary to achieve the desired goal has been chosen. The activity planning phase is thus time consuming because it is done by hand, and expensive because it requires domain expertise.

We are interested in building libraries of tasks in the situation calculus based action programming languages Golog [4] and its relatives. One motivation is the possibility of mitigating the time and cost overhead in complex activity planning by developing techniques that allow the reuse of libraries of tasks in an (ideally) automatic way. Although domain experts would still be required during the initial construction of such a library, once it contains a substantial amount of knowledge in the form of domain-specific, detailed tasks for achieving a number of goals, the cost incurred in this initial phase would be amortized over the many problems solved by reusing that knowledge. In order to achieve this, the library has to be in a form that can be reused by users who are not necessarily domain experts. For example, we would like the library to allow planning for the construction of a building on a soft clay ground, without consulting with an expert on how to lay foundations on such a ground if the library contains tasks for doing that. In this case, it should be enough to simply specify the type of ground, the type of building, and so on, in order to obtain from the library a detailed construction plan. If some details are missing or under specified, we ultimately would like the library to provide alternative plans, possibly including costs and risks.

Toward this end, we consider formalizing tasks and task execution in a stochastic, temporal version of Reiter's action theories in the sit-

uation calculus [6]. A task is defined by means of a set of axioms in the action theory, together with a Golog program. We also allow the user to specify properties (goals) that tasks are capable of bringing about, so that, given a goal, relevant tasks are executed without explicit invocation. This notion is useful for our purposes of designing a general task library that can be utilized by non expert users. Once this framework is in place, we then consider ways of generalizing goal achievement so that more of the burden required in retrieving information is shifted from the user to the knowledge base that contains the library. In particular, we consider the use of *generalized goals* so that tasks that achieve a very specific goal is also attempted on generalized versions of the specific goal. We also consider a form of generalized query based on *compatible types* of objects so that useful answers are obtained even in the case of incomplete information about domain objects' types. We have implemented this framework in Prolog building on the interpreters of the action programming languages mentioned above. The full version of this paper [3] contains a detailed description of this framework and sample runs obtained from the implementation.

2 Background

As a practical footing for our work, we consider a military operations planning problem described in [1]. Roughly, a problem instance in this domain consists of a description of a set of *tasks*, e.g. "anti-submarine operation", which are similar but more complex than actions in classical planning, together with a description of the initial state, including numerical quantities of available resources. A task description includes a duration, multiple possible outcomes with corresponding probabilities, resource requirements, and other pre-conditions. Also included are statements about goals that tasks can achieve, which are used to invoke relevant tasks as required by a goal. Moreover, some tasks require subgoals to hold prior or during their execution. These last two characteristics of tasks in this application resemble how Hierarchical Task Networks (HTNs) [2] and complex action languages like Golog operate. We appeal to the latter in this work, since these languages are based on a formal, logic foundation that allows stochastic, temporal actions, and is first-order [6].

One of the motivations for building a library of tasks is for using it in decision support. We are thus interested in query answering with respect to the library. For instance, we'd like to be able to issue queries such as: what is the prob. of achieving control of sea region $r2?$, and obtaining answers such as: A1: prob=0.8, executing tasks: aswWsubs,... A2: prob=0.75, executing tasks: aswWfrigates,... corresponding to two available ways of achieving the goal in the query.

3 Executing Tasks and Achieving Goals

As part of the specification of a task as described above, a "body" is defined in the form of a Golog program δ to be executed when the

¹ National ICT Australia and U. of New South Wales, Sydney, Australia, alfredo@cse.unsw.edu.au

task is invoked. A task is executed by 1) checking the arguments are of the right types, 2) substituting the arguments in the body of the task, and 3) executing the body. We formalize the execution of a task as the following Golog procedure:

```
proc execTsk(tsk,  $\vec{x}$ , t)
    typeCheck(tsk,  $\vec{x}$ )? ;
    ( $\exists \delta$ ).task(tsk,  $\vec{x}$ , t,  $\delta$ )? ;
     $\delta$ 
endProc
```

The definition of a task also includes declarations of what goals it can achieve. This allows requests that a goal be achieved without referring to a particular task. This is important for our purpose of designing libraries that can be used by non experts once the background knowledge has been stored in the libraries. For making such requests of achieving a goal, we define a procedure *achieve(goal, \vec{x} , t)*. In general there may be multiple tasks capable of achieving the same goal. For instance, the two tasks mentioned above, *aswWsubs* and *aswWfrigates*, represent two different ways of establishing control of a region of type “sea sub-surface”. Thus the *achieve* procedure is defined so that it non-deterministically chooses a task from among those that can achieve the goal, and executes it.

```
proc achieve(goal,  $\vec{x}$ , t)
    goal( $\vec{x}$ )? |
    ( $\pi tsk$ )[achievesG(tsk,  $\vec{x}$ , goal)? ;
    execTsk(tsk,  $\vec{x}$ , t)]
```

```
endProc
```

Notice that one of the choices for achieving the goal is to do nothing other than successfully test that the goal is already true. (πx) means choose an x , and $\delta_1|\delta_2$ means choose between executing δ_1 and δ_2 .

4 Generalized goals and achieve

A successful execution of a task, as defined above, requires that the parameters be of the correct type for the task. For instance, executing task *aswWsubs* on a region *reg* will only be successful if the knowledge base contains the fact that *reg* is of type *seaSubSurf*. Since we are interested in query answering we consider generalizing our framework so that goals and goal achievement can be organized into a hierarchy based on specificity.

Consider a scenario where a user issuing queries is interested in control of region *reg*, which he knows to be a sea region, but not whether it is a sea surface or a sea sub-surface. Suppose that the user does not know, but the info about the specific type of *reg* is in fact stored in the system. In such a situation, we would nevertheless like to be able to check if the goal *seaRegionCtrlEst* can be achieved for *reg*. The library should be able to invoke an appropriate task for *reg* based on its type, even if the user is not aware of it. However, we need to endow the system with the ability to figure out that, for example, if the more specific goal *seaSubSurfCtrlEst* is achieved, then the more general goal *seaRegionCtrlEst* has also been achieved. The fact that one goal is a generalized version of another cannot be derived from type information alone as it also depends on the nature of the goals. Thus for now we will rely on the knowledge engineer to provide additional information by means of statements of the form: *generalizedG(generalGoal, specificGoal)*. Since tasks that achieve a specific goal also achieve the less specific version of the goal, we can then define a generalization of *achievesG* capturing this relationship between goals by means of these statements and axiom:

$$\text{achievesG}(tsk, \vec{x}, g) \equiv \text{achievesG}_0(tsk, \vec{x}, g) \vee \\ \text{generalizedG}(g, g') \wedge \text{achievesG}_0(tsk, \vec{x}, g').$$

This results in the invocation of specific tasks, if necessary and available, for attempting to achieve a generalized goal. Type information plays a critical role in this.

Consider again the scenario mentioned above, but this time let us assume that the KB does not contain any info on whether *reg* is of type *seaSurf* or *seaSubSurf*. In this case, none of the specific tasks will execute since their type-check tests fail. Nevertheless, if the system knows *reg* is a sea region, then for a query whether control of *reg* can be achieved, we would like to obtain an answer that is more useful than “don’t know”. For instance, if a sea region must be either a *seaSurf* or a *seaSubSurf*, the system could give answers corresponding to these alternatives. For this, we introduce a test to check if the arguments passed to a task are *compatible* with the types required, where *compatible* is defined in terms of possible sub-types. Then we introduce execution of compatible tasks as a procedure *execCompTsk(tsk, \vec{x} , t)* similar to *execTsk(tsk, \vec{x} , t)*, but with a type-compatibility test instead of the strict type-check. Finally, we modify procedure *achieve* so that it tries compatible tasks when no specific task can execute:

```
proc achieve(goal,  $\vec{x}$ , t)
    goal( $\vec{x}$ )? |
    ( $\pi tsk$ )[achievesG(tsk,  $\vec{x}$ , goal)? ; execTsk(tsk,  $\vec{x}$ , t)] |
    [( $\forall tsk'$ )(achievesG(tsk',  $\vec{x}$ , goal)  $\supset$   $\neg$ typeCheck(tsk',  $\vec{x}$ ))? ; \\
    ( $\pi tsk$ )[achievesG(tsk,  $\vec{x}$ , goal)? ; \\
    execCompTsk(tsk,  $\vec{x}$ , t)]]
```

```
endProc
```

5 Conclusions

In this paper we present preliminary ideas on building a knowledge base containing a library of tasks for query answering. We discuss an encoding of stochastic, temporal tasks, and their execution, as complex actions in programming language Golog and its relatives. We consider how such a library can be used together with information about a domain’s object types to handle queries about the achievement of goals. We have also considered how to make the knowledge base more robust by allowing users to obtain useful answers even in cases where queries lack some degree of specificity, and where the knowledge base lacks detailed info about the types of certain domain objects. Much work remains to be done, including a proof of correctness of our implementation and extending the system to handle a larger class of queries.

REFERENCES

- [1] Douglas Aberdeen, Sylvie Thiébaut, and Lin Zhang, ‘Decision-theoretic military operations planning’, in *Procs. of ICAPS’04*, (2004).
- [2] Kutluhan Erol, James A. Hendler, and Dana S. Nau, ‘Complexity results for hierarchical task-network planning’, *Annals of Mathematics and Artificial Intelligence*, **18**, 69–93, (1996).
- [3] Alfredo Gabaldon, ‘Encoding Task Libraries in Complex Action Languages’, Avail. at <http://cse.unsw.edu.au/~alfredo/papers.html> (2006).
- [4] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl, ‘Golog: A logic programming language for dynamic domains’, *Journal of Logic Programming*, **31**(1–3), 59–83, (1997).
- [5] R. Reiter, ‘The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression’, in *Artificial Intelligence and Mathematical Theory of Computation*, ed., V. Lifschitz, 359–380, Academic Press, (1991).
- [6] Ray Reiter, *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*, MIT Press, (2001).

Automated Deduction for Logics of Default Reasoning

Laura Giordano^{*} and Valentina Gliozzi[§] and Nicola Olivetti[○] and Gian Luca Pozzato^{§ 1}

Abstract. We present a tableau calculus for the rational logic **R** of default reasoning, introduced by Kraus, Lehmann and Magidor. Our calculus is obtained by introducing suitable modalities to interpret conditional assertions, it makes use of labels to represent possible worlds, and it can be used to provide a decision procedure for **R**.

1 INTRODUCTION

The work [4] of Kraus, Lehmann and Magidor (KLM) proposed a formalization of nonmonotonic reasoning that leads to a classification of nonmonotonic consequence relations, determining a hierarchy of stronger and stronger systems. The so called *KLM properties* have been widely accepted as the “conservative core” of default reasoning; in the recent literature it is shown that many different semantics for plausible reasoning are characterized by the axioms of logic **R** (including κ -rankings, parametrized probabilistic structures, ϵ -semantics and possibilistic structures) [1].

In KLM logics, defeasible knowledge is assumed to be represented by a set of nonmonotonic conditionals or assertions of the form $A \succsim B$, whose reading is *normally (or typically) the A's are B's*. The operator “ \succsim ” is nonmonotonic, in the sense that $A \succsim B$ does not imply $A \wedge C \succsim B$. In this paper we focus on logic **R**, whose axiom system includes the rule of *rational monotonicity*: if $A \succsim B$ and $\neg(A \succsim \neg C)$ hold, then one can infer $A \wedge C \succsim B$. This rule allows a conditional to be inferred from a set of conditionals in absence of other information. For instance, a knowledge base K can contain the following set of conditional assertions: *adult* \succsim *work*, *adult* \succsim *taxpayer*, *student* \succsim *adult*, *student* \succsim \neg *work*, *student* \succsim \neg *taxpayer*, whose meaning is that adults typically work, adults typically pay taxes, students are typically adults, but they typically do not work, nor do they pay taxes. In rational logic **R** one can infer the following conditional assertions from the knowledge base K : *adult* \wedge *student* \succsim \neg *work* (giving preference to more specific information), *adult* \succsim \neg *student* (i.e. typical adults are not students). Moreover, if one further knows that *adult* $\not\sim$ *married* (i.e. it is not the case the adults are typically unmarried), one can also infer that *adult* \wedge *married* \succsim *work*. Observe that one cannot infer *student* \succsim *work*.

In this work we extend the investigation of tableau procedures for propositional KLM logics developed in [3] by considering the case of **R**. We present a tableau calculus for **R** which is sound, complete and terminating. Our approach is based on a novel interpretation of **R** into an extension of modal logic **G**. As a difference with [3], here we present a *labelled* tableau system.

¹ * Dip. di Informatica - Università del Piemonte Orientale A. Avogadro - Alessandria, Italy - e-mail: laura@mfn.unipmn.it
[§] Dip. di Informatica - Università degli Studi di Torino, Italy - e-mail: {gliozzi,pozzato}@di.unito.it
[○] LSIS - UMR CNRS 6168 Université Paul Cézanne (Aix-Marseille 3), France - e-mail: nicola.olivetti@univ.u-3mrs.fr

2 KLM RATIONAL LOGIC **R**

We briefly recall the semantics of **R**. The language of KLM logics consists just of conditional assertions $A \succsim B$. We consider a richer language allowing boolean combinations of conditional and propositional formulas. Our language \mathcal{L} is defined from a set of propositional variables ATM , the boolean connectives and the conditional operator \succsim . We use A, B, C, \dots to denote propositional formulas, whereas F, G, \dots are used to denote all formulas (even conditionals); Γ, Δ, \dots represent sets of formulas. The formulas of \mathcal{L} are defined as follows: if A is a propositional formula, $A \in \mathcal{L}$; if A and B are propositional formulas, $A \succsim B \in \mathcal{L}$; if F is a boolean combination of formulas of \mathcal{L} , $F \in \mathcal{L}$.

The semantics of **R** is defined by considering possible world structures with a preference relation (a strict partial order) $w < w'$ whose meaning is that w is preferred to w' . Moreover, the preference relation is supposed to be *modular*, i.e. for all w, w_1 and w_2 , if $w_1 < w_2$ then either $w_1 < w$ or $w < w_2$. We have that $A \succsim B$ holds in a model \mathcal{M} if B holds in all *minimal A-worlds* (w.r.t. $<$). This definition makes sense provided minimal *A*-worlds exist (whenever there are *A*-worlds). This is ensured by the *smoothness condition* in the next definition.

Definition 1 (Semantics of **R, Definition 14 in [5])** A rational model is a triple $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ where: \mathcal{W} is a non-empty set of items called worlds; $<$ is an irreflexive, transitive and modular relation on \mathcal{W} ; V is a function $V : \mathcal{W} \rightarrow \text{pow}(ATM)$, which assigns to every world w the set of atoms holding in that world. We define the truth conditions for a formula F as follows:

- If F is a boolean combination of formulas, $\mathcal{M}, w \models F$ is defined as for propositional logic;

- Let A be a propositional formula; we define $\text{Min}_{<}(A) = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A \text{ and } \forall w', w' < w \text{ implies } \mathcal{M}, w' \not\models A\}$;

- $\mathcal{M}, w \models A \succsim B$ if for all $w' \in \text{Min}_{<}(A)$ then $\mathcal{M}, w' \models B$.

The relation $<$ satisfies the smoothness condition: if $\mathcal{M}, w \models A$ then $w \in \text{Min}_{<}(A)$ or $\exists w' \in \text{Min}_{<}(A)$ such that $w' < w$. We say that a formula F is valid in a model \mathcal{M} ($\mathcal{M} \models F$), if $\mathcal{M}, w \models F$ for every $w \in \mathcal{W}$. A formula is valid if it is valid in every model \mathcal{M} .

Observe that from the modularity of $<$ follows that possible worlds of \mathcal{W} are *clustered* into equivalence classes, each class consisting of worlds that are incomparable to one another, with these classes being totally ordered. In other words the property of modularity determines a *ranking* of worlds so that the semantics of **R** can be specified equivalently in terms of *ranked* models [5].

3 THE TABLEAU CALCULUS FOR **R**

We present $\mathcal{T}\mathbf{R}$, a labelled tableau calculus for rational logic **R**. The calculus makes use of labels to represent possible worlds. We con-

sider a language \mathcal{L}_R and a denumerable alphabet of labels \mathcal{A} , whose elements are denoted by x, y, z, \dots . \mathcal{L}_R extends \mathcal{L} by formulas of the form $\Box A$, where A is propositional, whose intuitive meaning is as follows: $\Box A$ holds in a world w if A holds in all the worlds w' such that $w' < w$, that is to say:

Definition 2 (Truth condition of \square) $\mathcal{M}, w \models \square A$ if for every $w' \in \mathcal{W}$ if $w' < w$ then $\mathcal{M}, w' \models A$.

It is easy to see that \Box has (among others) the properties of the modal system G: the accessibility relation (defined as xRy if $y < x$) is transitive and does not have infinite ascending chains. From definition of $\text{Min}_<(A)$ in Definition 1 above, it follows that for any formula A , $w \in \text{Min}_<(A)$ iff $\mathcal{M}, w \models A \wedge \Box\neg A$. Our tableau calculus comprises two kinds of labelled formulas: (i) *world formulas* $x : F$, whose meaning is that F holds in the possible world represented by x ; (ii) *relation formulas* of the form $x < y$, where $x, y \in \mathcal{A}$, used to represent the relation $<$. We denote by $\alpha, \beta \dots$ world or relation formulas.

We define $\Gamma_{x \rightarrow y}^M = \{y : \neg A, y : \Box \neg A \mid x : \Box \neg A \in \Gamma\}$. The calculus \mathcal{TR} is presented in Figure 1.

We conclude this section by refining \mathcal{TR} in order to ensure termination. In general, non-termination in tableau calculi can be caused by two different reasons: 1. some rules copy their principal formula in the conclusion, so that they can be reapplied over the same formula without any control; 2. rules introducing *new* labels in their conclusion(s) can generate infinitely-many worlds, creating infinite branches. As far as \mathcal{TR} is concerned, one can prove the following Theorem:

Theorem 5 Given a set of formulas Γ , the tableau generated by TR for Γ only contains a finite number of labels.

This excludes the second source of non termination (point 2). Concerning point 1, \mathcal{TR} does not ensure a terminating proof search due to (\sim^+) , which can be applied without any control. We ensure the termination by putting some constraints on (\sim^+) in \mathcal{TR} . It is easy to observe that it is useless to apply the rule on the same conditional formula more than once by using the same label x . Indeed, all formulas in the premise of (\sim^+) are kept in the conclusions, then we can assume, without loss of generality, that two applications of (\sim^+) on x are consecutive. We observe that the second application is useless, since each of the conclusions has already been obtained after the first application, and can be removed. We prevent redundant applications of (\sim^+) by keeping track of labels (worlds) in which a conditional $u : A \vdash B$ has already been applied in the current branch. To this purpose, we add to each positive conditional a list of *used* labels, then we restrict the application of (\sim^+) only to labels not occurring in the corresponding list. Notice that also the rule $(<)$ copies its principal formula $x < y$ in the conclusion; however, this rule will be applied only a finite number of times. This is a consequence of the side condition of the rule application and the fact that the number of labels in a tableau is finite (Theorem 5). Hence the rule $(<)$ does not affect the termination of the calculus.

Theorem 6 (Termination of improved TR) Let Γ be a finite set of formulas, then any tableau generated by TR (modified as described above) is finite.

4 CONCLUSIONS

In this paper we have presented an analytic and terminating tableau calculus \mathcal{TR} for the rational logic \mathbf{R} . The paper is complementary to the work [3], where the other KLM systems are considered. In future research we plan to extend our proof-procedures to first-order KLM logics on the base of the analysis carried on in [2].

Acknowledgements. Part of this work has been supported by the projects “*PRIN05: Specification and verification of agent interaction protocols*” and “*PRIN2004: Logiche a piú valori e informazione incerta: metodologie algebriche e algoritmiche*”.

REFERENCES

- [1] N. Friedman and J. Y. Halpern, ‘Plausibility measures and default reasoning’, *Journal of the ACM*, **48**(4), 648–685, (2001).
 - [2] N. Friedman, J. Y. Halpern, and D. Koller, ‘First-order conditional logic for default reasoning revisited’, *ACM TOCL*, **1**(2), 175–207, (2000).
 - [3] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato, ‘Analytic Tableaux for KLM Preferential and Cumulative Logics’, in *Proc. of LPAR 2005*, volume 3835 of *LNAI*, pp. 666–681. Springer, (2005).
 - [4] S. Kraus, D. Lehmann, and M. Magidor, ‘Nonmonotonic reasoning, preferential models and cumulative logics’, *Artificial Intelligence*, **44**(1-2), 167–207, (1990).
 - [5] D. Lehmann and M. Magidor, ‘What does a conditional knowledge base entail?’, *Artificial Intelligence*, **55**(1), 1–60, (1992).

Figure 1. The calculus $\mathcal{T}\mathbf{B}$. To save space, rules for \rightarrow and \vee are omitted

Definition 3 (Truth conditions of formulas of TR) Given a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ and a labelled alphabet \mathcal{A} , we consider a mapping $I : \mathcal{A} \mapsto \mathcal{W}$. Given a formula α of the calculus TR , we define $\mathcal{M} \models_I \alpha$ as follows: $\mathcal{M} \models_I x : F$ iff $\mathcal{M}, I(x) \models F$; $\mathcal{M} \models_I x < y$ iff $I(x) < I(y)$. We say that a set Γ of formulas of TR is satisfiable if, for all formulas $\alpha \in \Gamma$, then $\mathcal{M} \models_I \alpha$, for some model \mathcal{M} and some mapping I .

A branch is a set of tableau formulas; a branch is closed if it contains an axiom, otherwise it is open. We say that a tableau is closed if all its branches are closed. The calculus \mathcal{TR} is sound and complete w.r.t. the semantics.

Theorem 4 (Soundness and completeness) Given a set of formulas Γ , there is a closed tableau for Γ iff Γ is unsatisfiable.

Reasoning about motion patterns

Björn Gottfried¹

Abstract. In order to analyse motion patterns for the purpose of making predictions and to explain the behaviour of systems methods are required for dealing with them. Difficulties in verbalising motion patterns for the purpose of communicating spatiotemporal situations, or in order to index spatiotemporal databases, indicate the importance of means which are simple to handle by human users. This paper proposes a set of sixteen atomic motion patterns which form the basis of a relation algebra. The relations are coarse but crisp, they allow imprecise knowledge about motion patterns to be dealt with, and they are easily accessible from the point of view of the user.

1 Atomic motion patterns

We are looking for a small graphical set of motion patterns, which can be obtained by simple observations and which can be drawn as a simple query-sketch, i.e. the representation we are looking for makes clear distinctions, not requiring sophisticated measurement tools. Additionally, motion patterns are to be described in a relative way between objects to avoid working with absolute positions.

In the most simple case we are concerned with two objects, O and P, each of which moves either towards the other one or away from it. An obvious way to represent such distinctions is the orientation grid which has been introduced by [3]. This reference system allows the relative positions between objects to be described with regard to both a left-right dichotomy and a towards-away dichotomy. Fig. 1 shows how this reference system is defined: an oriented reference line (depicted as an arrow) connects the positions of O and P.



Figure 1. Left: Two objects O and P define an arrow which connects them. Right: This arrow defines a two-dimensional reference system.

Then, there are the following possibilities:

- O moves towards P,
- O moves away from P,
- O moves left with respect to P, or
- O moves right with respect to P.

Combining these four relations simultaneously for both O and P we obtain $4^2 = 16$ relations which are depicted in Fig. 2. Each relation represents a motion pattern between two time points, t_0 and t_1 , and shows the way the objects take relative to each other during this time

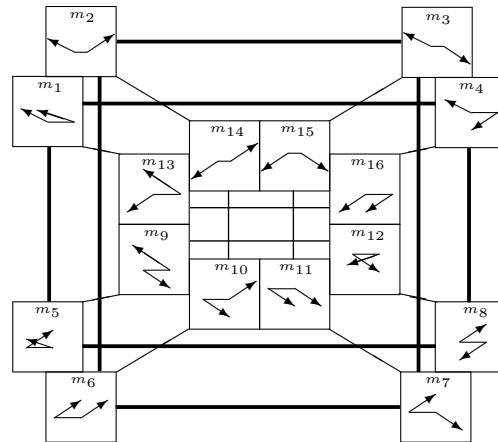


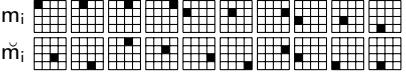
Figure 2. The neighbourhood graph of the 16 atomic motion patterns based on tripartite line tracks [1].

interval. While the two endpoints of the middle line define the initial positions of O and P at t_0 (as shown on the left hand side of Fig. 1), the head of the arrows (in Fig. 2) show their target positions at t_1 . Note that we confine ourselves to those situations in which both O and P are in general position at t_1 , i.e. we represent coarse tendencies. Including those situations in which objects can be observed precisely, so that it is possible to determine that they move exactly to the left or to the right, we obtain 36 relations. Including additionally those situations in which they can be observed to move precisely forward or backward we obtain 64 relations. Since such observations require precise measurement tools and since they are beyond what people can reliably observe it makes sense to deal with them as follows. We assign precise relations to other general relations, namely in the way that precisely forward equals forward to the right, precisely backward equals backward to the left, precisely left equals forward to the left, and precisely right equals backward to the right. However, if such precise distinctions matter they can be included in our formalism without changing any of our assumptions. In any case, we are concerned with a set of jointly exhaustive and pairwise disjoint relations. Similarly, we consider only objects which move and omit stationary objects. But stationary objects can be included in our formalism, extending only the set of atomic relations. Eventually, we include the identity relation, ID, which describes the movement pattern between two objects, such that they start at the same position and take exactly the same way. Graphically, this amounts to consider a single arrow.

Arranging the sixteen relations in a neighbourhood graph their similarity can be described by the conceptual distance among them. Fig. 2 shows this graph in which two relations are connected if they can be transformed into each other by continuously deforming or

¹ TZI, Universität Bremen, Germany, email: bg@tzi.de

m_1	m_2	m_3	m_4
m_5	m_6	m_7	m_8
m_9	m_{10}	m_{11}	m_{12}
m_{13}	m_{14}	m_{15}	m_{16}

m_i : 

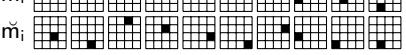
\check{m}_i : 

Figure 3. Left: iconic depiction of the relations; Right: table of converse relations (note that $\check{m}_i = m_i$; each relation that holds is printed in black).

moving the patterns without passing through another relation. Also, this graph aids in predicting motion patterns by constraining which motion patterns can occur next, provided that specific motion patterns are given. For instance, m_3 can directly follow m_2 , while m_8 cannot directly follow m_2 without passing other patterns in the meantime. The sixteen patterns form a set of jepd-relations on which a relation algebra can be defined [2]. Its table of converse relations and its composition table are given in Fig. 3 and Fig. 4, respectively. Furthermore, the usual set-theoretic operations apply. This allows the application of constraint based reasoning techniques.

2 An example application

We shall consider an example in which a claim is proved to be wrong in a civil hearing. Accordingly to a number of testimonies, O and P ran into opposite directions, away from each other. At the very same time P and Q separated, walking into different but not opposite directions, i.e. Q walked backwards with respect to P . Eventually, O and R walked towards each other, though on different sides of some object. Is it possible that R and Q is one and the same person, as claimed by a witness?

Formalising these evidences we obtain the following relations:

- $O \circ m_3 P$; though m_{14} would have also be a possible choice. However, one can choose between those relations. But as soon as we

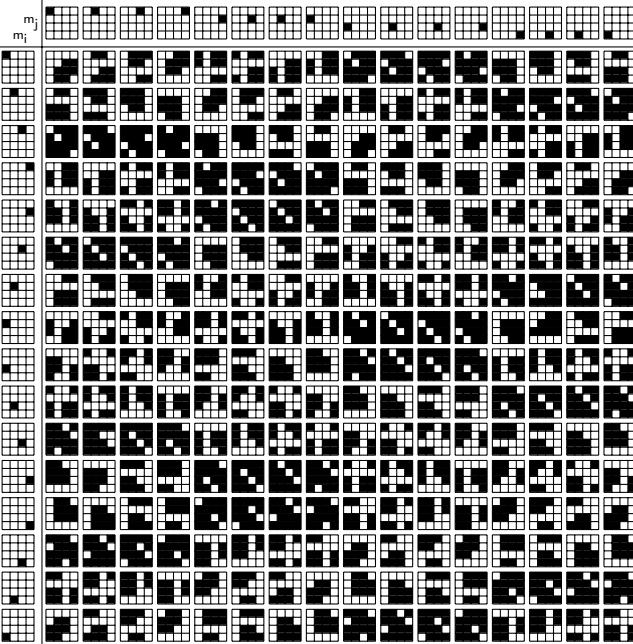
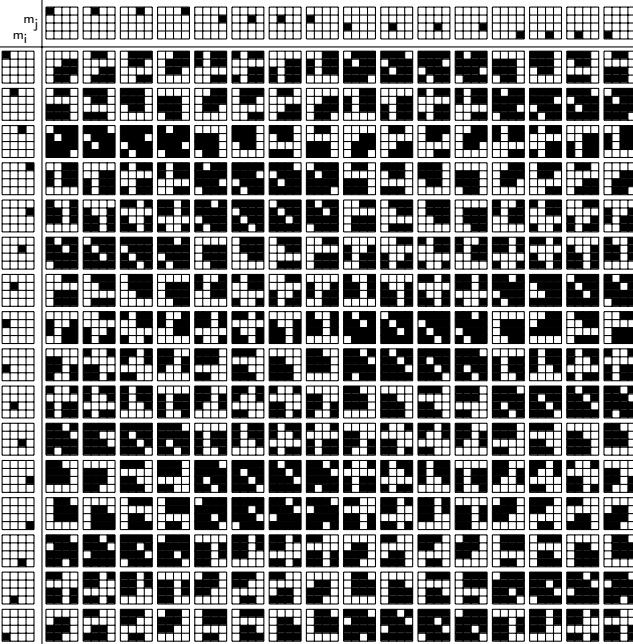
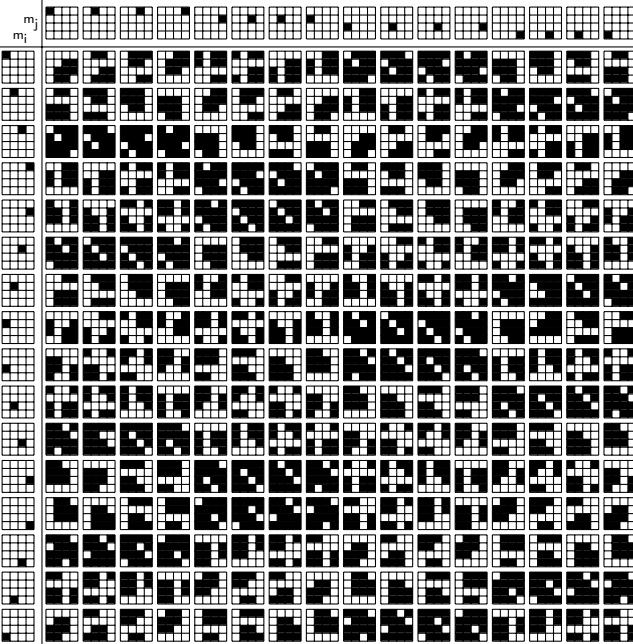
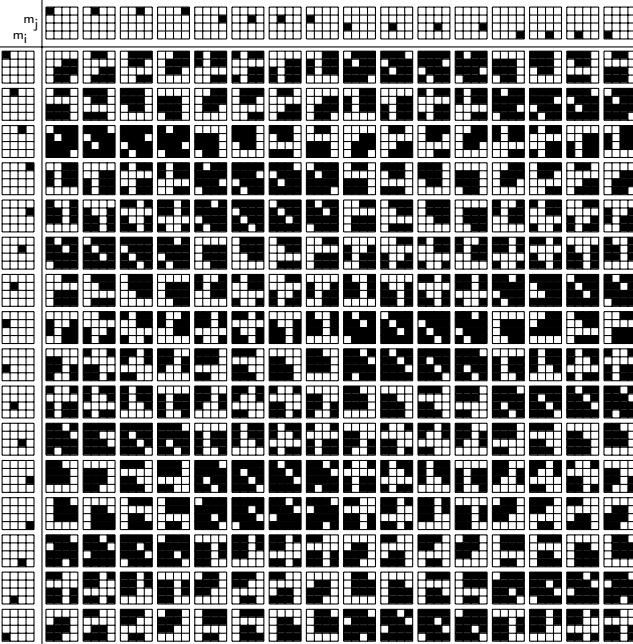
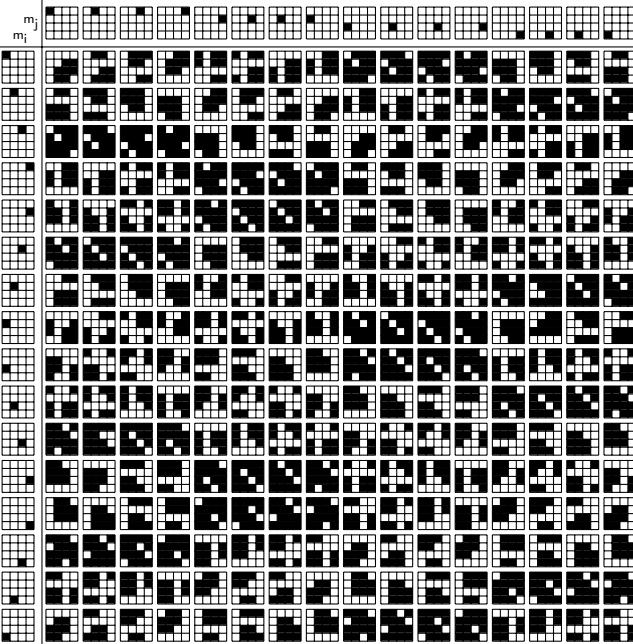
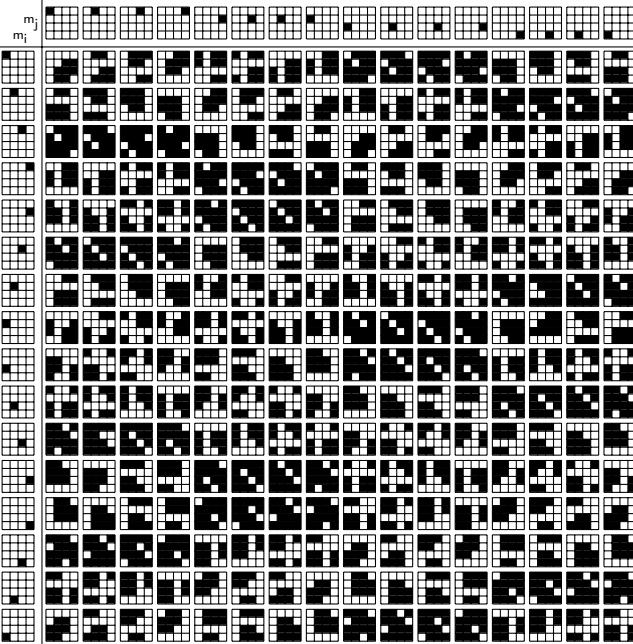
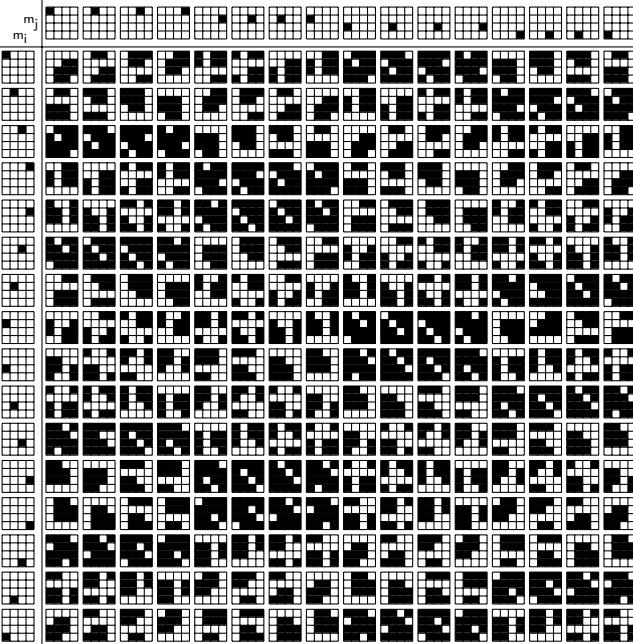
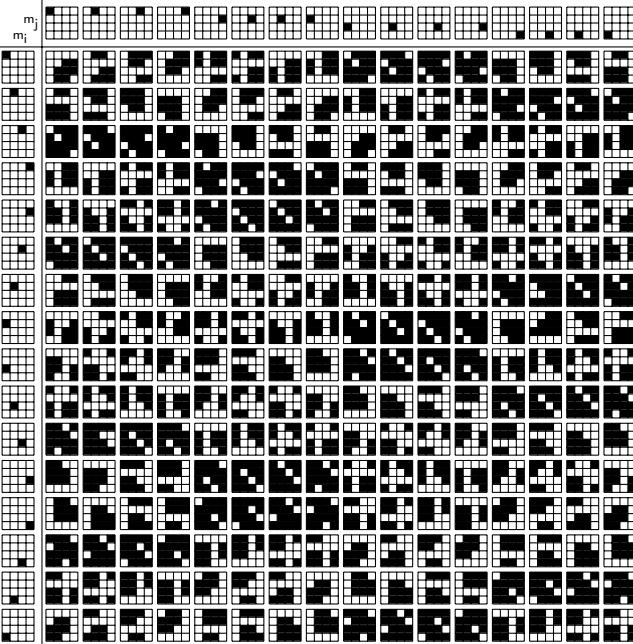
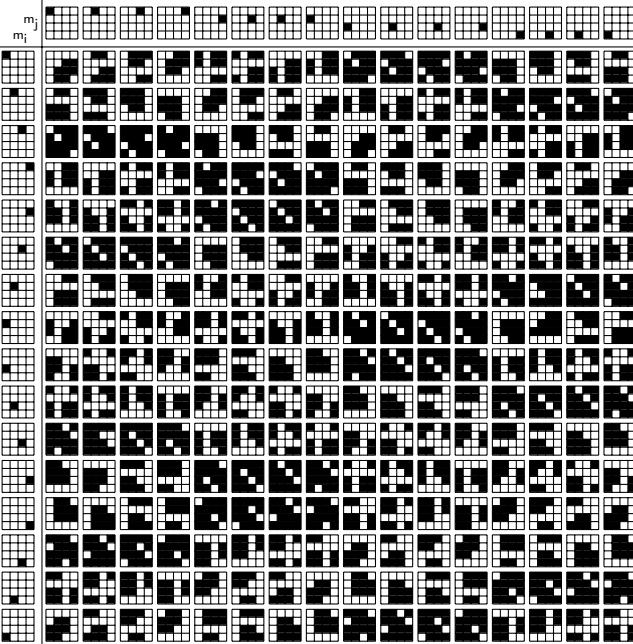
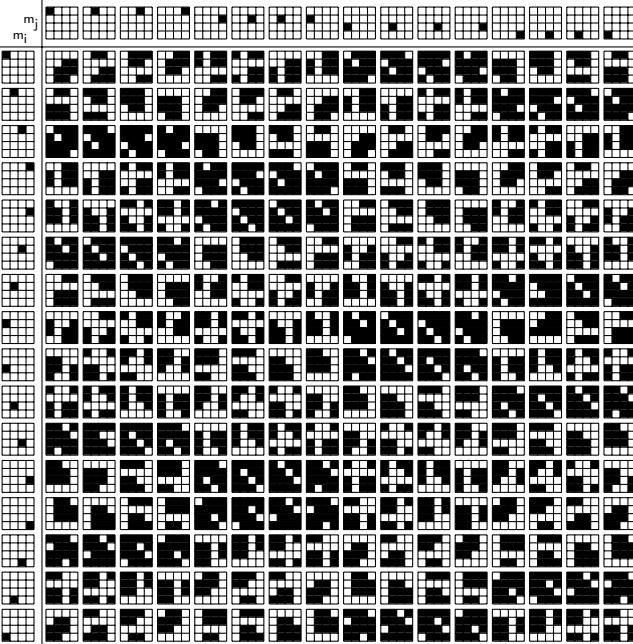
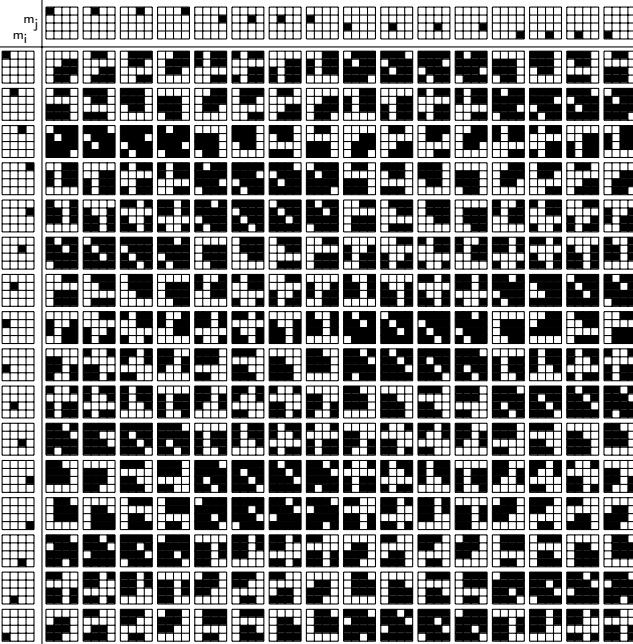
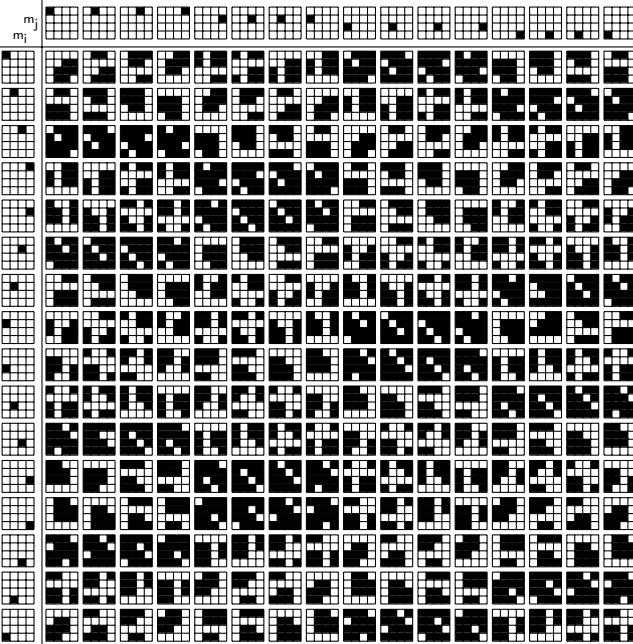
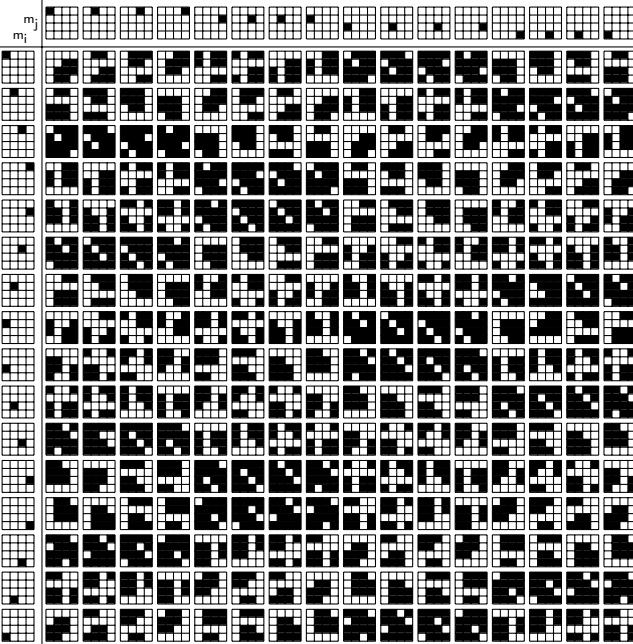
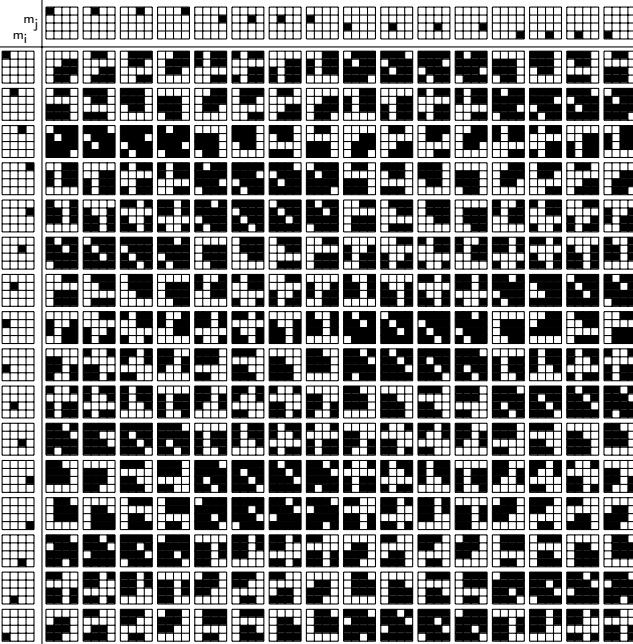
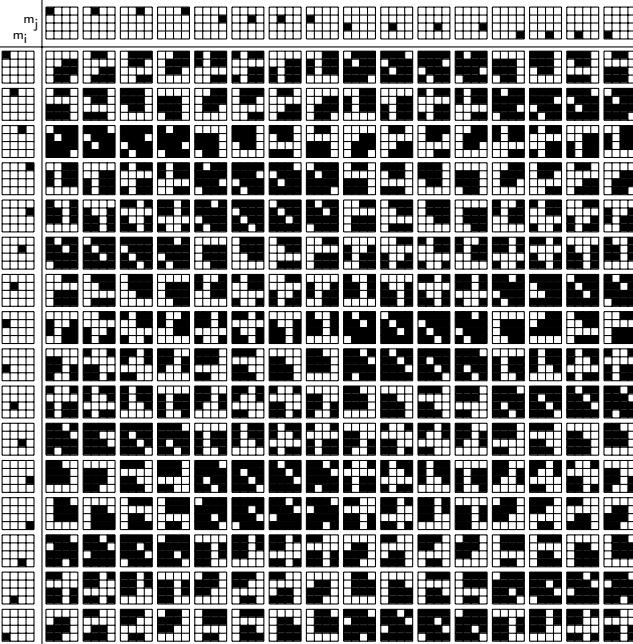
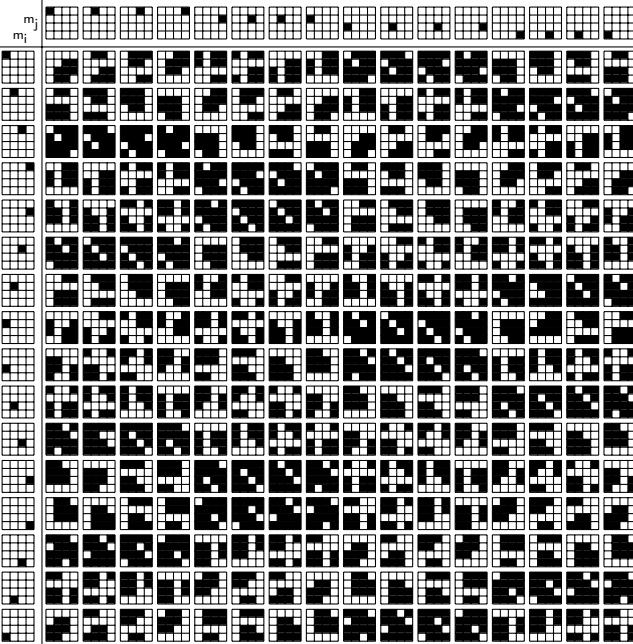
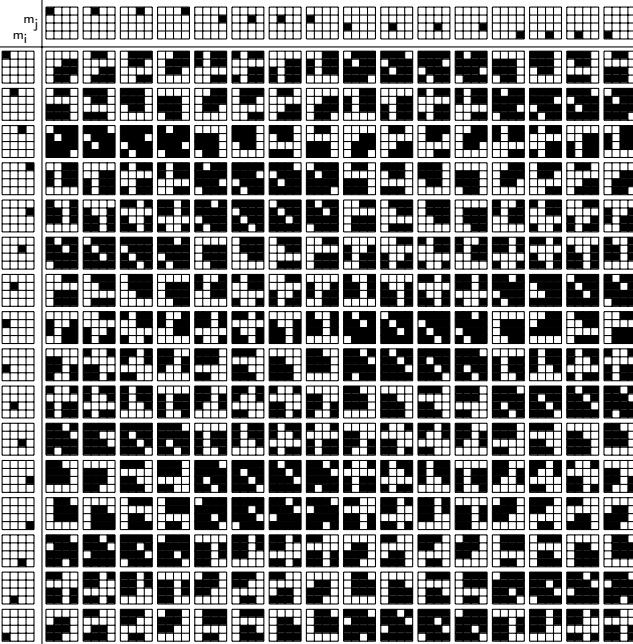
m_j	
m_1	
m_2	
m_3	
m_4	
m_5	
m_6	
m_7	
m_8	
m_9	
m_{10}	
m_{11}	
m_{12}	
m_{13}	
m_{14}	
m_{15}	
m_{16}	

Figure 4. Composition table: each entry shows the composition result in the form of an icon with the sixteen atomic motion patterns, which are arranged accordingly to the icon shown on the left hand side of Fig. 3; each relation that holds is printed in black.

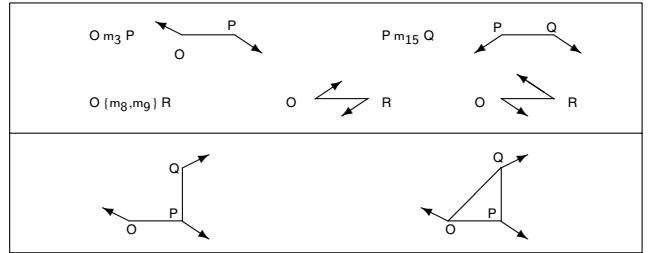


Figure 5. Upper part: graphical depiction of the observed motion patterns; lower part: the combination of $O \circ m_3 P$ and $P \circ m_{15} Q$ (left); in this instance it holds that $O \circ m_3 Q$.

commit ourselves to one of those relations, ensuing choices have to be made accordingly.

- $P \circ m_{15} Q$; here, we have to take m_{15} instead of m_2 since we decided for m_3 before.
- $O \{m_8, m_9\} R$

We assume that R and Q are equal. This especially implies that the relations between O and Q as well as between O and R have to be equal, too. We obtain the relation r between O and Q by composition: $O \circ Q = O \circ m_3 P \circ P \circ m_{15} Q = \{m_1, m_3, m_4, m_{11}, m_{12}, m_{13}, m_{15}, m_{16}\}$ (third row, fourteenth column in Fig. 4). As it holds that $\{m_8, m_9\} \not\subseteq r$, we conclude that R and Q cannot be the same person.

Fig. 5 illustrates this scenario. The assumptions are shown in the upper part of the Figure, while the lower part shows how these assumptions combine graphically. The right hand side of the lower part shows that it is neither possible to realise $O \circ m_8 Q$ nor $O \circ m_9 Q$. We conclude that R and Q are different persons.

3 Pro and con

- Whenever precise information is not available, coarse knowledge about motion patterns is better than to have no information at all.
- Incomplete information can be represented by sets of atomic motion patterns (that is, to mention those which might hold and to exclude those which cannot hold).
- Imprecise information is inherently represented by the atomic relations. Each one represents a range of variations, though perceptually clear distinctions are represented by different relations.
- As for other qualitative representations this approach suffers from a weak composition definition. But 43.75% of the relations possible can be excluded in the average in one inference step, as far as atomic motion patterns are used.
- While a set of sixteen qualitative atomic relations still exhibits a manageable size, including more relations (such as precisely left) the set of atomic relations becomes quite large, and as a consequence more difficult to handle in the context of adequate user interfaces (to observe and to draw precisely left). Note that it is sometimes just because of the difficulties to obtain precise information that such qualitative representations are used at all.

REFERENCES

- [1] B. Gottfried, ‘Tripartite Line Tracks, Qualitative Curvature Information’, in *COSIT 2003*, eds., W. Kuhn, M. Worboys, and S. Timpf, LNCS, 101–117, Springer, (2003).
- [2] P. Ladkin and R. Maddux, ‘On binary constraint problems’, *Journal of the Association for Computing Machinery*, **41**(3), 435–469, (1994).
- [3] K. Zimmermann and C. Freksa, ‘Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge’, *Applied Intelligence*, **6**, 49–58, (1996).

Applying OPRMs to Recursive Probability Models

Catherine Howard¹ and Markus Stumptner²

1 INTRODUCTION

Recursive probabilistic models(RPMs) are described as models where a variable may probabilistically depend on a potentially infinite, but finitely describable, set of variables [1,2]. In a RPM, a variable associated with a particular model entity may probabilistically depend on the same variable associated with a different model entity. Recursive models present two main challenges: how to compactly represent the model and how to perform inference. This paper discusses the application of the Object Oriented Probabilistic Relational Modelling Language (OPRML) to recursive probability models. We introduce the Iterative Junction Tree Construction (IJC) algorithm, a novel inference algorithm for OPRMs, which performs anytime inference to compute an approximate solution to the infinite BN described by a RPM in a finite time. We compare the performance of IJC against the Iterative Structured Variable Elimination (ISVE) algorithm [1] which was proposed for RPMs expressed using PRML. As part of this evaluation, we implemented ISVE, which to our knowledge, has not been previously implemented.

2 REPRESENTATION & REASONING

The OPRML is a typed relational language which is outlined in [3, 4]. Models produced using this language are called Object-Oriented Probabilistic Relational Models (OPRMs) [3, 4]. OPRMs specify a template for the probability distribution³ over a knowledge base (KB), where a knowledge base is defined as consisting of a set of OPRM classes and instances, a set of inverse statements and a set of instance statements. When representing and reasoning with RPMs, we place the following restrictions on the knowledge base:

- There are no inverse statements in the KB
- There are a finite number of named instances in the KB

The first restriction is used in this paper to simplify the discussion and can easily be lifted (see [1]). The second restriction cannot, because, as will be seen in Section 3, in order to reason with RPMs, we approximate the model using a finite network.

The OPRML can represent RPMs can represent recursive models very compactly for example, the OPRM shown in Figure 1 represents a RPM where a person's happiness depends on their Mother's happiness. While this model is simple, it is sufficient to illustrate the key points about representation and reasoning for RPMs.

Once we have the OPRM class specifications for the recursive model, we need an algorithm that will automatically construct the equivalent object oriented probabilistic network

¹Electronic Warfare and Radar Division, Defence Science and Technology Organisation, PO Box 1500, Edinburgh, South Australia, 5111, email: catherine.howard@dsto.defence.gov.au

²Advanced Computing Research Centre, University of South Australia, Adelaide, South Australia, 5095. email: mst@cs.unisa.edu.au

³ As with traditional BN, the probability model specified by a OPRM must be acyclic.

from the information stored in these class specifications and perform inference. Howard and Stumptner [4] present the Knowledge Based Model Construction (KBMC) and Junction Tree Construction (JC) algorithms which automatically construct models from OPRM KBs. However, these algorithms are unsuitable for RPMs because the set of variables in the recursive model is potentially infinite, these model construction algorithms may never terminate.

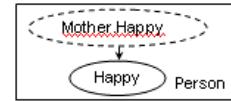


Figure 1. The OPRM for the Happiness model.

The Iterative JC (IJC) algorithm is an anytime algorithm which is an extension of the JC algorithm which addresses this problem. The IJC embodies the solution quality/execution time tradeoff by incrementally building a larger, more complex finite model to *approximate* the infinite network described by the RPM. After each iteration, the model becomes a better and better approximation and as this *approximate model* describes the probability distribution over a finite number of variables, the model construction process will terminate and exact inference can be performed in a finite time.

At design time, the class specifications are used to create the equivalent object oriented probabilistic network for each class in the KB. This network is then translated directly into a 'local' junction tree. An interface clique is created which consists of all nodes in the class's interface, κ . This interface clique is connected into the local junction tree and any loops created during this process are removed. Thus at design time, each class is 'precompiled' into a local junction tree and these local junction trees are stored in a cache.

When the IJC algorithm is executed, the desired order of approximation, n_D , is specified. Then, starting with an order of approximation, n , of one and increasing n by one in each iteration, the algorithm builds an increasing complex model which approximates the infinite network described by the RPM. In each iteration, the model from the previous iteration is built upon by instantiating the required number of generic unnamed instances of the Person class. The number of instances that are required to be added to the model will be specified by the probabilistic dependencies within the class. For example, in the Phenome and Intelligence models, a person's Phenome and Intelligence attributes depend on Phenome and Intelligence attributes of *both* parents, so 2^n generic instances will be added to the model in each iteration (where n is the order of approximation which equals the iteration number), while for the Happiness model, only 1^n instances will be added to the model per iteration.

Whenever a generic unnamed instance, Z , of the Person class is created, the appropriate local rooted junction tree, JT_{RZ} , is instantiated. A root clique for the model is created which contains all the nodes in all the generic unnamed instances' interfaces, κ_Z . Each local instance junction tree is then connected to the root clique, ω_R , to create the 'global junction

tree'. Exact inference can then be performed using this global junction tree using standard junction tree message passing techniques to return the approximate probability distribution over the query variables.

So the n^{th} iteration of the IJC algorithm produces a model which is the n^{th} order approximation to the infinite BN described by the RPM. In our example models, the order of approximation equates to the number of generations of people included in the model. As n (and hence execution time) increases, the algorithm includes more and more generations in the approximate model and returns a better and better approximate solution to the query. IJC is an interruptible anytime algorithm: it can be interrupted at any time and will return the best computed approximation so far. The meta-level control paradigm employed by the IJC algorithm is to allocate the desired order of approximation and allow the algorithm to construct a model of this order, regardless of how long it takes.

3 EXPERIMENTS

Both ISVE and IJC were implemented in MATLAB augmented with the BN Toolbox (BNT) and executed using a Pentium III, 1.2GHz computer.

Figure 2 shows that for small orders of approximation ($n \leq 15$), the IJC performs slightly better than ISVE. However, after this, there is a blowout in the inference time which can be seen clearly in Figure 2a. This blowout is clearly due to inference time as shown in Figure 2d. The inference time is being effected by the major limitation of the IJC algorithm: as the root clique must contain all the interface nodes of all the generic unnamed instances currently in the model, the root clique becomes very large very quickly. If instead of creating a root node, we propagate 'fill ins' between interface nodes in a manner similar to Multiply Sectioned Bayesian Networks, this problem may be ameliorated.

In order to compare the performance of IJC and ISVE, we need to be able to define objective metrics for measuring the quality of the solution provided by the algorithms at any given time. We cannot use the Kullback-Leibler Entropy which is the traditional measure for the quality of an approximate algorithm because as RPMs can potentially include an infinite number of variables, the true probability distribution over the query variables can not be calculated. The behaviour of an anytime algorithm is often described using a performance profile (PP) which describes how the quality of the algorithm output varies as a function of execution time. Figure 3 shows the PP of the ISVE and IJC algorithms for the Happiness model for a query on the Happy attribute where the quality is defined as one minus the normalized rate of change of the probability distribution over the query variable. This figure shows that the quality of both ISVE and IJC are non-decreasing functions of execution time. Figure 3b shows the rate of change in the probability distribution over the query variable for both algorithms. This plot shows that the rate of change of the quality of the solution is large at the early stages of execution and decreases over time. As can be seen from these figures, the IJC algorithm reaches a steady state in a shorter execution time.

Both ISVE and IJC approximate the infinite BN by a finite network which is incrementally built up to the desired order of approximation which is set at run time. Both algorithms then use exact inferencing techniques on these resulting model approximations to obtain a probability distribution over the query variables. ISVE uses a structured form of variable elimination while IJC uses a message passing algorithm. Both

IJC and ISVE take advantage of the structure of the domain by encapsulating information within individual objects so that only a small amount of information needs to be passed between objects. And both algorithms take advantage of reuse.

However, unlike PRMs, the OPRML can be used to represent RPMs without placing any restrictions on the language. We have conducted similar experiments using different domain models and we have found that, as you would expect, the comparative performance of IJC and ISVE is model dependant.

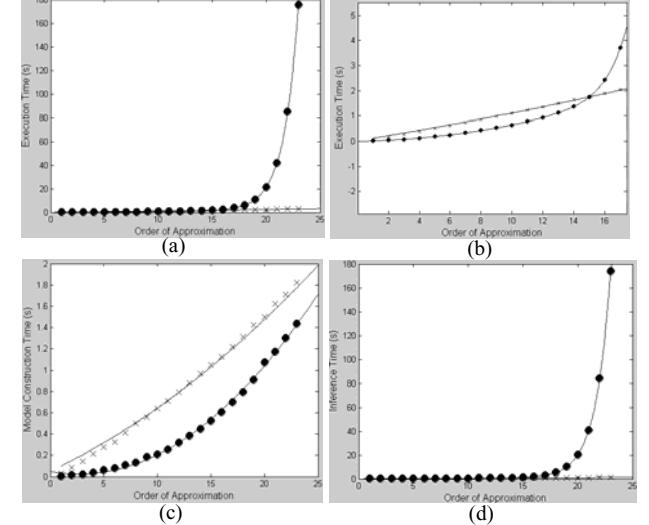


Figure 2. Plots of the IJC (circles) and ISVE (crosses) (a) execution (c) model construction and (d) inference times for the Happiness model. Fig. 2(b) is an expansion of the $n=1$ through to $n=17$ section of plot (a).

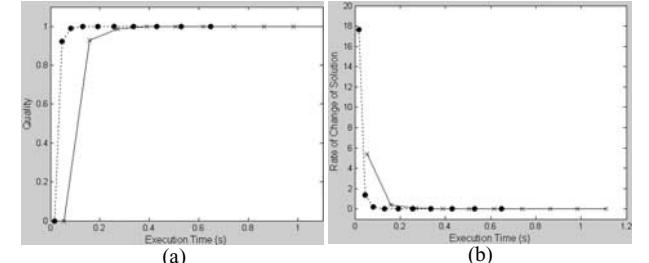


Figure 3. The quality of the IJC (circles) and ISVE (crosses) algorithms for the Happiness model.

4 REFERENCES

1. Pfeffer, A. and D. Koller. *Semantics and Inference for Recursive Probability Models*. in *National Conference on Artificial Intelligence (AAAI)*. 2000.
2. Pfeffer, A.J., *Probabilistic Reasoning for Complex Systems*, in *Department of Computer Science*. 1999, Stanford University. p. 304.
3. Howard, C. and M. Stumptner. *Situation Assessments Using Object Oriented Probabilistic Relational Models*. in *Proc. 8th Int'l Conference on Information Fusion*. 2005. Philadelphia, USA.
4. Howard, C. and M. Stumptner. *Model Construction Algorithms for Object Oriented Probabilistic Relational Models*. (to appear) *19th International Florida Artificial Intelligence Research Society's Conference*. 2006. (FLAIRS 06). Melbourne Beach, Florida.

Variable forgetting in preference relations over propositional domains

Philippe Besnard¹ and Jérôme Lang¹ and Pierre Marquis²

Abstract. Representing (and reasoning about) preference relations over combinatorial domains is computationally expensive. For many problems involving such preferences, it is relevant to simplify them by projecting them on a subset of variables. We investigate several possible definitions, focusing without loss of generality on propositional (binary) variables.

1 Introduction and background

Decision-making problems are concerned with managing agents' preferences. When the set of alternatives is small, preferences can be represented explicitly, by simply ranking alternatives, and the associated tasks such as optimization or aggregation are computationally easy. However, in many real-world applications, domains have a combinatorial structure. In that case, managing agents' preferences can be an enormous computational burden. This has led to the study of compact preference representation languages.

For some problems it may be relevant to process a preference relation, so as to simplify it and make it more compact, even if this results in a loss of information. Especially, it may be helpful to *project* a preference relation on a subset of the variables. This way of summarizing a preference relation is relevant in particular when some variables are more important than others, or when some variables should be assigned prior to others. Consider for instance a group decision making scenario. Rather than aggregating the whole preference relations before finding out an optimal assignment of variables, which generally is computationally intractable, it may be a good idea to focus on "primary" variables first, project the preference relation on those variables, aggregate them, decide on the values to be assigned to those variables, and only then consider secondary variables.

Projection operations have not been considered much as far as preference relations are concerned, but there is a huge amount of work about projecting (or *marginalizing*) probability distributions (especially when they are represented by Bayesian networks), and more generally valuation functions [6, 3], as well as sets of constraints, and propositional formulas (cf. the forgetting operation [5]). In this paper we define similar projection operations for ordinal preference relations. For the sake of simplicity, we focus on combinatorial domains formed from propositional (i.e., binary) variables.

Let V be a finite set of propositional variables. For any subset X of V , an X -alternative is an element of 2^X , that

¹ IRIT, UPS, 31062 Toulouse, France; {besnard, lang}@irit.fr

² CRIL, U. d'Artois, 62307 Lens, France; marquis@cril.univ-artois.fr

is, an assignment of a binary truth value to each one of the variables in X . X -alternatives are denoted by \vec{x} , \vec{x}' etc. If X and Y are disjoint subsets of V then the concatenation of $\vec{x} \in 2^X$ and $\vec{y} \in 2^Y$ is the $X \cup Y$ -alternative, denoted by $\vec{x}\vec{y}$, assigning values to variables of X (resp. Y) as \vec{x} (resp. \vec{y}) does.

A V -preference relation R is a reflexive and transitive relation over 2^V . The *strict preference* $>_R$ associated with R is the strict order defined by $\vec{v} >_R \vec{v}'$ iff $R(\vec{v}, \vec{v}')$ and not $R(\vec{v}', \vec{v})$. The *indifference relation* \sim_R associated with R is the equivalence relation defined by $\vec{v} \sim_R \vec{v}'$ iff $R(\vec{v}, \vec{v}')$ and $R(\vec{v}', \vec{v})$. If neither $R(\vec{v}, \vec{v}')$ nor $R(\vec{v}', \vec{v})$ holds, then \vec{v} and \vec{v}' are *incomparable* w.r.t. R , denoted by $\vec{v}Q\vec{v}'$. For the sake of notation, when we specify a preference relation explicitly, we omit pairs coming from reflexivity and transitivity. R^* denotes the transitive closure of a relation R over 2^V .

For any V -preference relation R and any partition $\{X, Y, Z\}$ of V , X is preferentially independent from Y given Z w.r.t. R iff for all $\vec{x}, \vec{x}' \in 2^X$, all $\vec{y}, \vec{y}' \in 2^Y$ and all $\vec{z} \in 2^Z$, $R(\vec{x}\vec{y}\vec{z}, \vec{x}'\vec{y}'\vec{z})$ implies $R(\vec{x}\vec{y}'\vec{z}, \vec{x}'\vec{y}'\vec{z})$. If $Z = \emptyset$ then we say that X is preferentially independent from $V \setminus X$ w.r.t. R .

2 Lower and upper projections

Definition 1 (lower and upper projections) Let R be a V -preference relation and $X \subseteq V$. Let $Y = V \setminus X$.

- $R_L^{\downarrow X}$, called the lower projection of R on X , is the binary relation over X defined as follows: $R_L^{\downarrow X}(\vec{x}, \vec{x}')$ holds iff $R(\vec{x}\vec{y}, \vec{x}'\vec{y})$ holds for all $\vec{y} \in 2^Y$;
- $R_U^{\downarrow X}$, called the upper projection of R on X , is the transitive closure of the relation R' over X defined by:
 $R'(\vec{x}, \vec{x}')$ holds iff $R(\vec{x}\vec{y}, \vec{x}'\vec{y})$ holds for some $\vec{y} \in 2^Y$.

When R is complete, $R_U^{\downarrow X}$ is obviously complete as well but $R_L^{\downarrow X}$ may fail to be complete. Other properties are:

Proposition 1 For any V -preference relations R , R' and any $X, Y \subseteq V$:

1. $R_L^{\downarrow X}$ and $R_U^{\downarrow X}$ are X -preference relations;
2. if $R \subseteq R'$ then $R_L^{\downarrow X} \subseteq (R')^{\downarrow X}_L$ and $R_U^{\downarrow X} \subseteq (R')^{\downarrow X}_U$;
3. $(R \cap R')^{\downarrow X}_L = R_L^{\downarrow X} \cap (R')^{\downarrow X}_L$; $(R \cap R')^{\downarrow X}_U \subseteq R_U^{\downarrow X} \cap (R')^{\downarrow X}_U$;
4. $((R \cup R')^*)^{\downarrow X}_U = (R_U^{\downarrow X} \cup (R')^{\downarrow X}_U)^*$;
 $(R \cup R')^{\downarrow X}_L \supseteq (R_L^{\downarrow X} \cup (R')^{\downarrow X}_L)^*$;
5. $(R_L^{\downarrow X})^{\downarrow Y}_L = (R_L^{\downarrow Y})^{\downarrow X}_L$; $(R_U^{\downarrow X})^{\downarrow Y}_U = (R_U^{\downarrow Y})^{\downarrow X}_U$.

Proposition 2 For any V -preference relation R and any $X \subseteq V$, $R_L^{\downarrow X} = R_U^{\downarrow X}$ iff X is preferentially independent from $V \setminus X$ w.r.t. R .

3 Optimistic and pessimistic projections

Definition 2 (optimistic/pessimistic projections) Let R be a V -preference relation and $X \subseteq V$. Let $Y = V \setminus X$.

- $R_{StrongOpt}^{\downarrow X}$, the strong optimistic projection of R on X , is defined by: $R_{StrongOpt}^{\downarrow X}(\vec{x}, \vec{x}') \text{ iff } \exists \vec{y} \forall \vec{y}', R(\vec{x}\vec{y}, \vec{x}'\vec{y}');$
- $R_{WeakOpt}^{\downarrow X}$, the weak optimistic projection of R on X , is defined by: $R_{WeakOpt}^{\downarrow X}(\vec{x}, \vec{x}') \text{ iff } \forall \vec{y}' \exists \vec{y} R(\vec{x}\vec{y}, \vec{x}'\vec{y}');$
- $R_{StrongPess}^{\downarrow X}$, the strong pessimistic projection of R on X , is defined by: $R_{StrongPess}^{\downarrow X}(\vec{x}, \vec{x}') \text{ iff } \exists \vec{y}' \forall \vec{y}, R(\vec{x}\vec{y}, \vec{x}'\vec{y}');$
- $R_{WeakPess}^{\downarrow X}$, the weak pessimistic projection of R on X , is defined by: $R_{WeakPess}^{\downarrow X}(\vec{x}, \vec{x}') \text{ iff } \forall \vec{y} \exists \vec{y}' R(\vec{x}\vec{y}, \vec{x}'\vec{y}').$

The optimistic projections focus on finding some possibility to have \vec{x} dominating \vec{x}' whatever the context for \vec{x}' . The pessimistic projections focus on finding some possibility to have \vec{x}' dominated by \vec{x} whatever the context for \vec{x} .

When R is complete, $R_{StrongOpt}^{\downarrow X}$ and $R_{WeakOpt}^{\downarrow X}$ coincide, as well as $R_{StrongPess}^{\downarrow X}$ and $R_{WeakPess}^{\downarrow X}$, and all four are complete. In this case, $R_{StrongOpt}^{\downarrow X}(\vec{x}, \vec{x}')$ (and equivalently $R_{WeakOpt}^{\downarrow X}(\vec{x}, \vec{x}')$) iff the best alternatives extending \vec{x} are at least as good as the best alternatives extending \vec{x}' , whereas $R_{StrongPess}^{\downarrow X}(\vec{x}, \vec{x}')$ (and equivalently $R_{WeakPess}^{\downarrow X}(\vec{x}, \vec{x}')$) iff the worst alternatives extending \vec{x} are at least as good as the worst alternatives extending \vec{x}' .³

Proposition 3 We have the following inclusions:

- $R_L^{\downarrow X} \subseteq R_{StrongOpt}^{\downarrow X} \subseteq R_{WeakOpt}^{\downarrow X} \subseteq R_U^{\downarrow X};$
- $R_L^{\downarrow X} \subseteq R_{StrongPess}^{\downarrow X} \subseteq R_{WeakPess}^{\downarrow X} \subseteq R_U^{\downarrow X}.$

4 Examples

R_1	R_2	R_3	R_4	R_5
			xy	
xy	$x\bar{y}$	$xy \rightarrow x\bar{y}$	\downarrow	$xy \sim x\bar{y}$
\downarrow	\downarrow		$\bar{x}\bar{y}$	\downarrow
$\bar{x}y$	$\bar{x}\bar{y}$	$\bar{x}y \leftarrow \bar{x}\bar{y}$	$\swarrow \searrow$	$\bar{x}y \sim \bar{x}\bar{y}$
		$\bar{x}y$	\downarrow	
			$\bar{x}y$	

When $R = R_1$: all projections on x coincide and are equal to the preference relation $x > \bar{x}$. All projections of R_1 on y coincide and are equal to the preference relation $yQ\bar{y}$ in which y and \bar{y} are incomparable.

When $R = R_2$: all projections on x are equal to $xQ\bar{x}$. $R_L^{\downarrow \{y\}}$ as well as $R_{StrongOpt}^{\downarrow \{y\}}$ and $R_{StrongPess}^{\downarrow \{y\}}$ are equal to $yQ\bar{y}$, while $R_U^{\downarrow \{y\}}$ as well as $R_{WeakOpt}^{\downarrow \{y\}}$ and $R_{WeakPess}^{\downarrow \{y\}}$ are equal to $y \sim \bar{y}$.

When $R = R_3$: $R_L^{\downarrow \{x\}}$ is equal to $xQ\bar{x}$; $R_U^{\downarrow \{x\}}$ is equal to $x \sim \bar{x}$; $R_{StrongOpt}^{\downarrow \{x\}}$ and $R_{WeakOpt}^{\downarrow \{x\}}$ are equal to $x > \bar{x}$; $R_{StrongPess}^{\downarrow \{x\}}$ is equal to $xQ\bar{x}$, while $R_{WeakPess}^{\downarrow \{x\}}$ is equal to $x \sim \bar{x}$. Things are symmetric for the projections on y .

When $R = R_4$: all projections on x are equal to $x > \bar{x}$. $R_L^{\downarrow \{y\}}$ is equal to $yQ\bar{y}$; $R_U^{\downarrow \{y\}}$ is equal to $y \sim \bar{y}$; the optimistic

projections on y (which coincide because R is complete) are equal to $y > \bar{y}$; the pessimistic projections on y (which coincide, again because R is complete) are equal to $\bar{y} > y$.

When $R = R_5$, all projections on x are equal to $x > \bar{x}$ and all projections on y are equal to $y \sim \bar{y}$.

5 Connection to propositional logic

Let L_V be the propositional language built up from V . When $\varphi \in L_V$, $Var(\varphi)$ denotes the set of propositional variables occurring in φ . We make use of the next two notions, close to the ones considered in [4]: let $\varphi \in L_V$ and $X \subseteq V$, the *strongest necessary condition* of φ on X , denoted by $\exists(V \setminus X).\varphi$, is the logically strongest formula ψ of L_X s.t. $\varphi \models \psi$; the *weakest sufficient condition* of φ on X , denoted by $\forall(V \setminus X).\varphi$, is the logically weakest formula ψ of L_X s.t. $\psi \models \varphi$. $\exists(V \setminus X).\varphi$ is also known as the *forgetting* of $V \setminus X$ in φ .

A V -preference relation is *bipartite* iff there exists $G \subseteq 2^V$ such that for all $\vec{v}, \vec{v}' \in 2^V$, then $R(\vec{v}, \vec{v}')$ holds iff $\vec{v} \in G$ or $\vec{v}' \in 2^V \setminus G$; a *characteristic formula* θ_R of a bipartite V -preference relation R is a propositional formula whose set of models is exactly G (in symbols, $Mod(\theta_R) = G$). Note that if R is bipartite, it is complete and then strong and weak projections coincide.

Proposition 4 Let R be a bipartite V -preference relation and a characteristic formula θ_R of it. Let $X \subseteq V$ and $Y = V \setminus X$. Then

- $R_{WeakOpt}^{\downarrow X} = R_{StrongOpt}^{\downarrow X}$ is the bipartite relation whose characteristic formula is $\exists(V \setminus X).\theta_R$.
- $R_{WeakPess}^{\downarrow X} = R_{StrongPess}^{\downarrow X}$ is the bipartite relation whose characteristic formula is $\forall(V \setminus X).\theta_R$.

6 Conclusion and perspectives

This paper is meant to pave the way towards simplifying and decomposing preference relations over combinatorial structures. It is still a preliminary work and raises many questions. One of the most salient issues that we did not investigate is about computing the various notions of projection when the initial preference relation is represented in a *compact representation language*. The long version of the paper also includes a section on the various possible notions of *independence* of a preference relation from a set of variables.

REFERENCES

- [1] R. Brafman and M. Tennenholtz. Modeling agents as qualitative decision makers. *Artificial Intelligence*, 94:217–268, 1997.
- [2] D. Dubois and H. Prade. Possibility theory as a framework for qualitative decision theory. In *Proceedings of IJCAI-95*, pages 1924–1930, 1995.
- [3] J. Kohlas and P. Shenoy. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5, chapter Computation in valuation algebras, pages 5–39. Kluwer Academic Publishers, 2000.
- [4] F. Lin. On the strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128:143–159, 2001.
- [5] F. Lin and R. Reiter. Forget it! In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159, New Orleans, 1994.
- [6] P. Shenoy. A valuation-based language for expert systems, international journal of approximate reasoning. *International Journal of Approximate Reasoning*, 3(2):383–411, 1989.

³ These criteria are reminiscent of those used in qualitative decision theory (see e.g., [1, 2] – with the slightly different interpretation that X -alternatives represent possible decisions and elements of $(V \setminus X)$ -alternatives represent possible states of the world).

Two Orthogonal Biases for Choosing the Intensions of Emerging Concepts in Ontology Refinement

Francesca A. Lisi and Floriana Esposito¹

1 INTRODUCTION

Ontology Refinement is the adaptation of an existing ontology to a specific domain or the needs of a particular user [5]. In this paper we consider the problem of finding subconcepts of a known concept C_{ref} , called *reference concept*, in the existing ontology Σ in the light of new knowledge coming from a data source Π . We assume that a concept C consists of two parts: an *intension* $int(C)$ and an *extension* $ext(C)$. The former is an expression belonging to a logical language \mathcal{L} whereas the latter is a set of objects that satisfy the former. More formally, given

- a reference concept $C_{ref} \in \Sigma$,
- a data set $\mathbf{r} = \Sigma \cup \Pi$,
- a language \mathcal{L}

our **Ontology Refinement problem** is to find a directed acyclic graph (DAG) \mathcal{G} of concepts C_i such that (i) $int(C_i) \in \mathcal{L}$ and (ii) $ext(C_i) \subset ext(C_{ref})$. Note that C_{ref} is among both the concepts defined in Σ and the symbols of \mathcal{L} . Furthermore $ext(C_i)$ relies on a notion of satisfiability of $int(C_i)$ w.r.t. \mathbf{r} . Note that \mathbf{r} includes Σ because in Ontology Refinement, as opposite to other forms of Ontology Learning such as Ontology Extraction (or Building), it is mandatory to consider the existing ontology and its existing connections.

A **Knowledge Representation and Reasoning (KR&R) framework** that turns out to be suitable for our problem is the one offered by the *hybrid system* $\mathcal{AL}\text{-log}$ [2]. It allows for the specification of both relational and structural data: the former is based on DATALOG [1], the latter on \mathcal{ALC} [8]. The integration of the two logical formalisms is provided by the so-called constrained DATALOG clause, i.e. a DATALOG clause with variables possibly constrained by concepts expressed in \mathcal{ALC} . Within this KR&R framework, the data set \mathbf{r} is represented as a $\mathcal{AL}\text{-log}$ knowledge base \mathcal{B} and the language \mathcal{L} contains expressions, called $\mathcal{O}\text{-queries}$, of the form

$$Q = q(X) \leftarrow \alpha_1, \dots, \alpha_m \& X : C_{ref}, \gamma_1, \dots, \gamma_n,$$

relating individuals of C_{ref} to individuals of other concepts (*task-relevant concepts*) also occurring in Σ . Thus, for a concept C , $int(C)$ is an \mathcal{O} -query $Q \in \mathcal{L}$ and $ext(C)$ is the set $answerset(Q, \mathcal{B})$ of correct answers to Q w.r.t. \mathcal{B} . The DAG \mathcal{G} is structured according to the *subset relation* between concept extensions.

As a **solution approach** to the problem, we follow a recent trend in Cluster Analysis: using *frequent (association) patterns* as candidate clusters [10]. A frequent pattern is an intensional description, expressed in a language \mathcal{L} , of a subset of a given data set \mathbf{r} whose cardinality exceeds a user-defined threshold (*minimum support*). Note

that patterns can refer to multiple levels of description granularity (*multi-grained patterns*). In any case, a frequent pattern highlights a regularity in \mathbf{r} , therefore it can be considered as the clue of a data cluster. In our context these clues are called *emerging concepts* because they are concepts whose only extension is determined. We propose to discover emerging concepts by following the approach of [3] to frequent pattern discovery at l , $1 \leq l \leq maxG$, levels of description granularity and k , $1 \leq k \leq maxD$, levels of search depth. It adapts [6] to the case of a pattern space of \mathcal{O} -queries organized according to the quasi-order $\succeq_{\mathcal{B}}$ (\mathcal{B} -subsumption).

Since multiple patterns can have the same set of supporting individuals, an **open issue** is to determine the intension for each emerging concept. As a solution to this issue we propose the criterion of choice illustrated in the next Section.

2 THE CHOICE CRITERION

The choice criterion for concept intensions has been obtained by combining two orthogonal biases: a language bias and a search bias. A *bias* concerns anything which constrains the search for theories [7]. In particular, *language bias* has to do with restrictions on the clauses in the search space whereas *search bias* has to do with the way a system searches its space of permitted clauses. In our context, the language bias allows the user to define conditions on the form of \mathcal{O} -queries to be accepted as concept intensions. E.g., it is possible to state which is the minimum level of description granularity and whether (all) the variables must be ontologically constrained or not. The search bias allows the user to define a preference criterion based on \mathcal{B} -subsumption. More precisely, it is possible to state whether the *most general description* (*m.g.d.*) or the *most specific description* (*m.s.d.*) w.r.t. $\succeq_{\mathcal{B}}$ has to be preferred. Since $\succeq_{\mathcal{B}}$ is not a total order, it can happen that two patterns P and Q , belonging to the same language \mathcal{L} , can not be compared w.r.t. $\succeq_{\mathcal{B}}$. In this case, the m.g.d. (resp. m.s.d.) of P and Q is the union (resp. conjunction) of P and Q . The two biases are combined as follows. For each frequent pattern $P \in \mathcal{L}$ that fulfills the language bias specification, the procedure for building the DAG \mathcal{G} from the set $\mathcal{F} = \{\mathcal{F}_k^l \mid 1 \leq l \leq maxG, 1 \leq k \leq maxD\}$ checks whether a concept C with $ext(C) = answerset(P)$ already exists in \mathcal{G} . If one such concept is not retrieved, a new node C with $int(C) = P$ and $ext(C) = answerset(P)$ is added to \mathcal{G} . Note that the insertion of a node can imply the reorganization of the DAG to keep it compliant with the subset relation on extents. If the node is already present in \mathcal{G} , its intension is updated according to the search bias specification.

As an illustration, we report the results of two experiments conducted on the $\mathcal{AL}\text{-log}$ knowledge base \mathcal{B}_{CIA} that has been obtained

¹ Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, 70125 Bari, Italy, email: {lisi, esposito}@di.uniba.it

by adding DATALOG facts² extracted from the on-line 1996 CIA World Fact Book³ to an \mathcal{ALC} ontology Σ_{CIA} concerning the concepts Country, EthnicGroup, Language, and Religion. The parameter settings are: $C_{\text{ref}} = \text{MiddleEastCountry}$, $\max D = 5$, $\max G = 3$, $\min sup^1 = 20\%$, $\min sup^2 = 13\%$, and $\min sup^3 = 10\%$. Both experiments start from the same set \mathcal{F} of 53 frequent patterns out of 99 candidate patterns and require the descriptions to have all the variables ontologically constrained by concepts from the second granularity level on. When the m.g.d. criterion is adopted, the procedure of graph building returns twelve concepts among which:

- $\mathcal{C}'_2 \in \mathcal{F}_3^2$
 $q(A) \leftarrow \text{speaks}(A, B) \&$
 $A:\text{MiddleEastCountry}, B:\text{AfroAsiaticLanguage}$
 $\{\text{IR, SA, YE}\}$
- $\mathcal{C}'_3 \in \mathcal{F}_3^2$
 $q(A) \leftarrow \text{speaks}(A, B) \&$
 $A:\text{MiddleEastCountry}, B:\text{IndoEuropeanLanguage}$
 $\{\text{ARM, IR}\}$
- $\mathcal{C}'_4 \in \mathcal{F}_5^2$
 $q(A) \leftarrow \text{speaks}(A, B), \text{believes}(A, C) \&$
 $A:\text{MiddleEastCountry},$
 $B:\text{AfroAsiaticLanguage}, C:\text{MonotheisticReligion}$
 $\{\text{IR, SA}\}$
- $\mathcal{C}'_6 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, 'Druze') \& A:\text{MiddleEastCountry}$
 $\{\text{IL, SYR}\}$
- $\mathcal{C}'_7 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, B) \&$
 $A:\text{MiddleEastCountry}, B:\text{JewishReligion}$
 $\{\text{IR, IL, SYR}\}$
- $\mathcal{C}'_8 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, B) \&$
 $A:\text{MiddleEastCountry}, B:\text{ChristianReligion}$
 $\{\text{ARM, IR, IRQ, IL, JOR, RL, SYR}\}$
- $\mathcal{C}'_9 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, B) \&$
 $A:\text{MiddleEastCountry}, B:\text{MuslimReligion}$
 $\{\text{BRN, IR, IRQ, IL, JOR, KWT, RL, OM, Q, SA, SYR, TR, UAE}\}$

organized in the DAG $\mathcal{G}'_{\text{CIA}}$. They are numbered according to the chronological order of insertion in $\mathcal{G}'_{\text{CIA}}$ and annotated with information of the generation step. From a qualitative point of view, concepts \mathcal{C}'_2 ⁴ and \mathcal{C}'_9 well characterize Middle East countries. Armenia (ARM), as opposite to Iran (IR), does not fall in these concepts. It rather belongs to the weaker characterizations \mathcal{C}'_3 and \mathcal{C}'_8 . This proves that our procedure performs a 'sensible' clustering. Indeed Armenia is a well-known borderline case for the geo-political concept of Middle East, though the Armenian is usually listed among Middle Eastern ethnic groups. Modern experts tend nowadays to consider it as part of Europe, therefore out of Middle East. But in 1996 the on-line CIA World Fact Book still considered Armenia as part of Asia.

When the m.s.d. criterion is adopted, the intensions for the concepts \mathcal{C}'_4 , \mathcal{C}'_6 and \mathcal{C}'_7 change as follows:

- $\mathcal{C}'_4 \in \mathcal{F}_5^2$
 $q(A) \leftarrow \text{speaks}(A, B), \text{believes}(A, C) \&$
 $A:\text{MiddleEastCountry},$
 $B:\text{ArabicLanguage}, C:\text{MuslimReligion}$
 $\{\text{IR, SA}\}$
- $\mathcal{C}'_6 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, 'Druze'), \text{believes}(A, B),$
 $\text{believes}(A, C), \text{believes}(A, D) \&$
 $A:\text{MiddleEastCountry}, B:\text{JewishReligion},$
 $C:\text{ChristianReligion}, D:\text{MuslimReligion}$
 $\{\text{IL, SYR}\}$
- $\mathcal{C}'_7 \in \mathcal{F}_3^3$
 $q(A) \leftarrow \text{believes}(A, B), \text{believes}(A, C), \text{believes}(A, D) \&$
 $A:\text{MiddleEastCountry}, B:\text{JewishReligion},$
 $C:\text{ChristianReligion}, D:\text{MuslimReligion}$
 $\{\text{IR, IL, SYR}\}$

In particular \mathcal{C}'_6 and \mathcal{C}'_7 look quite overfitted to data. Yet overfitting allows us to realize that what distinguishes Israel (IL) and Syria (SYR) from Iran is just the presence of Druze people.

3 CONCLUSIONS AND FUTURE WORK

Our approach to Ontology Refinement takes an ontology as input and returns subconcepts of one of the concepts in the ontology. This is done by discovering strong associations between concepts in the input ontology. The idea of applying association rules in Ontology Learning has been already investigated in [4]. Yet there are several differences: [4] is conceived for Ontology Extraction instead of Ontology Refinement, uses generalized association rules (bottom-up search) instead of multi-level association rules (top-down search), adopts propositional logic instead of First Order Logic (FOL). Also our work has contact points with Vrain's proposal [9] of a top-down incremental but distance-based method for conceptual clustering in a mixed object-logical representation. For the future we plan to extensively evaluate our approach. Experiments will show, among the other things, how emerging concepts depend on the minimum support thresholds set for the stage of frequent pattern discovery.

References

- [1] S. Ceri, G. Gottlob, and L. Tanca, *Logic Programming and Databases*, Springer, 1990.
- [2] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf, 'AL-log: Integrating Datalog and Description Logics', *Journal of Intelligent Information Systems*, **10**(3), 227–252, (1998).
- [3] F.A. Lisi and D. Malerba, 'Inducing Multi-Level Association Rules from Multiple Relations', *Machine Learning*, **55**, 175–210, (2004).
- [4] A. Maedche and S. Staab, 'Discovering Conceptual Relations from Text', in *Proc. of the 14th European Conf. on Artificial Intelligence*, ed., W. Horn, pp. 321–325. IOS Press, (2000).
- [5] A. Maedche and S. Staab, 'Ontology Learning', in *Handbook on Ontologies*, eds., S. Staab and R. Studer, Springer, (2004).
- [6] H. Mannila and H. Toivonen, 'Levelwise search and borders of theories in knowledge discovery', *Data Mining and Knowledge Discovery*, **1**(3), 241–258, (1997).
- [7] S.-H. Nienhuys-Cheng and R. de Wolf, *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*, Springer, 1997.
- [8] M. Schmidt-Schauss and G. Smolka, 'Attributive concept descriptions with complements', *Artificial Intelligence*, **48**(1), 1–26, (1991).
- [9] C. Vrain, 'Hierarchical conceptual clustering in a first order representation.', in *Foundations of Intelligent Systems*, eds., Z.W. Ras and M. Michalewicz, volume 1079 of *LNAI*, 643–652, Springer, (1996).
- [10] H. Xiong, M. Steinbach, A. Ruslim, and V. Kumar, 'Characterizing pattern based clustering', Tech. Report TR 05-015, Dept. of Computer Science and Engineering, University of Minnesota, USA, (2005).

² <http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-rel-facts.flp>

³ <http://www.odci.gov/cia/publications/factbook/>

⁴ \mathcal{C}'_2 is less populated than expected because \mathcal{B}_{CIA} does not provide facts on the languages spoken for all countries.

Computing possible and necessary winners from incomplete partially-ordered preferences

M. S. Pini*, F. Rossi*, K. B. Venable*, T. Walsh**¹

1 INTRODUCTION

There are many situations where we wish to represent and reason with preferences. We consider here how to combine the preferences of multiple agents despite incompleteness and incomparability in their preference orderings. An agent's preference ordering may be incomplete because, for example, we are in the middle of eliciting their preferences. It may also contain incomparability since, for example, we might have multiple criteria we wish to optimize.

To combine preferences, we use *social welfare functions*, which map a *profile*, that is, a sequence of partial orders (one for each agent), to a partial order (the result). For example, the Pareto social welfare function orders A before B iff every agent orders A before B , else if there is some disagreement between agents declares A and B to be incomparable.

Since agents' preferences may be incomplete, we consider all the possible ways they can be completed. In each possible completion, we may obtain different optimal elements (or *winners*). This leads to the idea of *possible winners* (those outcomes which are winners in at least one possible completion) and *necessary winners* (those outcomes which are winners in all possible completions) [5].

Possible and necessary winners are useful in many scenarios including preference elicitation [3]. In fact, elicitation is over when the set of possible winners coincides with that of the necessary winners [4]. In addition, as we argue later, preference elicitation can focus just on the incompleteness concerning those outcomes which are possible and necessary winners. We can ignore completely all other outcomes.

Whilst computing the sets of possible and necessary winners is in general a difficult problem, we identify sufficient conditions where we can obtain the necessary winners and an upper approximation of the set of possible winners in polynomial time. Such conditions concern either the language for stating preferences, or general properties of the preference aggregation function.

2 FROM THE COMBINED RESULT TO WINNERS

We would like to compute efficiently the set of possible and necessary winners, as well as to determine whether a given outcome is a possible or a necessary winner. In general, even if the social welfare function is polynomially computable, incompleteness in the profile may require us to consider an exponential number of completions. As observed in [5], determining the possible winners is in NP, and the necessary winners is in coNP.

¹*: Department of Pure and Applied Mathematics, University of Padova, Italy. Email: {mpini,frossi,kvenable}@math.unipd.it. **: NICTA and UNSW, Sydney, Australia. Email: tw@cse.unsw.edu.au

We consider therefore a compact representation of all the completions that is polynomial in size. This necessarily throws away information by compacting together results into a single combined result. Given a social welfare function f and a possibly incomplete profile ip , we consider a complete binary graph, whose nodes are the outcomes, and whose arcs are labeled by non-empty subsets of $\{<, >, =, \bowtie\}$, where \bowtie represents incomparability. Label l is on the arc between outcomes A and B if there exists a completion in which A and B are related by l in the result. We call this structure the *combined result* of f and ip and we denote it with $cr(f, ip)$. We first consider how to compute the possible and necessary winners given the combined result. We will then consider how to compute the combined result.

Consider the arc between an outcome A and an outcome B in the combined result. Then, if this arc has the label $A < B$, A is not a necessary winner, since there is an outcome B which is better than A in some result. If this arc *only* has the label $A < B$, then A is not a possible winner since we must have $A < B$ in all results. Moreover, consider all the arcs between A and every other outcome C . Then, if no such arc has label $A < C$, then A is a necessary winner. Notice, however, that, even if none of the arcs connecting A have just a single label $A < C$, then we cannot be sure that A is a possible winner: A could be better than some outcomes in every completion, but there might be no completion where it is better than all of them. We define the following algorithm to compute the necessary winners and a superset of the possible winners.

Algorithm 1: Necessary and possible winners

```

Input:  $\Omega$ : set of outcomes;  $f$ : preference aggregation function;
 $ip$ : incomplete profile;
Output:  $P, N$ : sets of outcomes;
 $P \leftarrow \Omega$ ;
 $N \leftarrow \Omega$ ;
foreach  $O \in \Omega$  do
    if  $\exists O' \in \Omega$  such that  $(O < O') \in cr(f, ip)$  then
         $N \leftarrow N - O$ ;
    if  $\exists O' \in \Omega$  such that  $(O < O') \in cr(f, ip)$  and  $(OrO') \notin cr(f, ip)$  for  $r \in \{=, >, \bowtie\}$  then  $P \leftarrow P - O$ ;
return  $P, N$ ;

```

If NW is the set of necessary winners and PW is the set of possible winners, Algorithm 1 obtains $N = NW$ and $P = PW^*$, which is a superset of the set of possible winners, in time quadratic in the number of outcomes. This is optimal as the combined result we are starting from is itself quadratic in size. PW^* can be larger than the set of possible winners for two reasons. First, since we consider one arc at a time, we may not recognize completions which violate tran-

sitivity. Second, we start from the combined result where we have already thrown away some information.

The first reason for approximation (that is, non-transitivity) can be eliminated. In fact, given an outcome O , we eliminate $O < O'$ from the label of each arc connecting O in the combined result, and test whether the new structure, which we call the *possibility structure* of outcome O (or $\text{poss}(O)$) is consistent with transitivity. This test is equivalent to testing the consistency of a set of branching temporal constraints [2], which is NP-hard. Fortunately, however, there are many classes of branching temporal constraint problems which are tractable [2], and these are likely to occur when combining preferences. For example, one tractable class is defined by restricting the labels to the set $\{<, >, =\}$. That is, we do not permit incomparability (\bowtie) in the result. Another tractable case is when we combine preferences using the Pareto social welfare function.

3 TRACTABLE COMPUTATION OF POSSIBLE AND NECESSARY WINNERS

Unfortunately, a naive computation of the combined result requires applying the social welfare function to an exponential number of completions. We identify here some properties often held by preference aggregation functions which allow us to compute an upper approximation to the combined result in polynomial time. This upper approximation can then be used to compute possible and necessary winners again in polynomial time. Let us denote the set of labels of an arc between A and B in the combined result as $\text{rel}(A, B)$.

The first property we consider is *independence to irrelevant alternatives (IIA)*. A social welfare function is said to be IIA if, for any pair of outcomes A and B , the ordering between A and B in the result depends only on the relation between A and B given by the agents. Many preference aggregation functions are IIA, and this is a desirable property related to the notion of fairness in voting theory [1].

Given a function which is IIA, to compute the set $\text{rel}(A, B)$, we just need to ask each agent its preference over the pair A and B , and then use f to compute all possible results between A and B . However, if agents have incompleteness between A and B , f has to consider all the possible completions, which is exponential in the number of such agents.

Assume now that f is also *monotonic*. We say that an outcome B improves with respect to another outcome A if the relationship between A and B does not move left along the following sequence: $>, \geq, (\bowtie \text{ or } =), \leq, <$. A social welfare function f is monotonic if, given any two profiles p and p' and any two outcomes A and B , if passing from p to p' B improves with respect to A in one agent i and $p_j = p'_j$ for all $j \neq i$, then in passing from $f(p)$ to $f(p')$ B improves with respect to A .

To compute $\text{rel}(A, B)$ assuming IIA and monotonicity, we just need to consider the agents' preferences over the pair A and B . However, because of monotonicity, we don't need to consider all possible completions for all agents with incompleteness between A and B , but just two completions: $A < B$ and $B > A$. Function f will return a result for each of these two completions, say AxB and AyB , where $x, y \in \{<, >, =, \bowtie\}$. Since f is monotonic, the results of all the other completions will necessarily be between x and y in the ordering $>, \geq, (\bowtie \text{ or } =), \leq, <$.

By taking all such relations, we obtain a superset of $\text{rel}(A, B)$, that we call $\text{rel}^*(A, B)$. In fact, monotonicity of f assures that, if we consider profile $A < B$ and we get a certain result, then considering profiles where A is in a better position w.r.t. B (i.e., $A > B$, $A = B$,

or $A \bowtie B$), will give an equal or better situation for A in the result.

Notice that we have obtained the set $\text{rel}^*(A, B)$ in time polynomial in the number of agents as we only needed to consider two completions. Under the IIA and monotonicity assumptions, we can thus obtain in polynomial time a labeled graph similar to the combined result, but with possibly more labels on the arcs. Then, we can apply the same reasoning as in the previous section to this labeled graph. In fact, we can show that the additional labels do not change the necessary and possible winners computed by the algorithm. So, under the IIA and monotonicity assumptions, we can obtain NW and PW^* in polynomial time.

4 PREFERENCE ELICITATION

One use of necessary and possible winners is in eliciting preferences. At each stage in preference elicitation, there is a set of possible and necessary winners. When $NW = PW$, preference elicitation can be stopped since we have enough information to declare the winners, no matter how the remaining incompleteness is resolved [4]. At the beginning, NW is empty and PW contains all outcomes. As preferences are declared, NW grows and PW shrinks. At each step, an outcome in PW can either pass to NW or become a loser.

In those steps where PW is still larger than NW , we can use these two sets to guide preference elicitation and avoid useless work. In fact, to determine if an outcome $A \in PW - NW$ is a loser or a necessary winner, it is enough to ask agents to declare their preferences over all pairs involving A and another outcome, say B , in PW . Any outcome outside PW is a loser, and thus is dominated by at least one possible winner.

If the preference aggregation function is IIA, then all those pairs (A, B) with a defined preference for all agents can be avoided, since they will not help in determining the status of outcome A . Moreover, IIA allows us to consider just one profile when computing the relations between A and B in the result, and assures that the result is a precise relation, that is, either $<$, or $>$, or $=$, or \bowtie . In the worst case, we need to consider all such pairs. To determine all the winners, we thus need to know the relations between A and B for all $A \in PW - NW$ and $B \in PW$.

During preference elicitation, we can also use the consistency test defined in the previous section to test the consistency of the preferences of each agent. In particular, if the agent declares outcomes to be ordered or incomparable, testing the consistency of the agents' preferences is tractable. If the consistency test is successful, we can exploit the information deduced by the consistency enforcement to avoid asking for preferences which are implied by previously elicited ones. If instead we detect inconsistency, then we can help the agent to make their preferences consistent by providing one or more triangles where consistency fails.

REFERENCES

- [1] K. J. Arrow, A. K. Sen, and K. Suzumura, *Handbook of Social Choice and Welfare.*, North-Holland, Elsevier, 2002.
- [2] M. Broxvall and P. Jonsson, 'Point algebras for temporal reasoning: Algorithms and complexity', *Artificial Intelligence*, **149**(2), 179–220, (2003).
- [3] L. Chen and P. Pu, 'Survey of preference elicitation methods', Technical Report IC/200467, Swiss Federal Institute of Technology in Lausanne (EPFL), (2004).
- [4] V. Conitzer and T. Sandholm, 'Vote elicitation: Complexity and strategy-proofness', in *Proc. AAAI/IAAI 2002*, pp. 392–397, (2002).
- [5] K. Konczak and J. Lang, 'Voting procedures with incomplete preferences', in *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, (2005).

What's a head without a body?

Christian Anger¹ and Martin Gebser¹ and Tomi Janhunen² and Torsten Schaub^{1,3}

Abstract. Concepts in Answer Set Programming (ASP) are normally defined in terms of atoms. We show that the treatment of atoms and bodies (of rules) as equitable computational objects may yield exponential speed-ups, even for standard ASP-solvers such as *smodels*. To this end, we give simple transformations providing solvers with access to both kinds of objects and show that some families of examples can be solved exponentially faster after they have been transformed. We prove that these transformations may yield exponentially smaller search spaces.

1 SIMULATING HEADS AND BODIES

For a detailed introduction to Answer Set Programming (ASP), we refer the reader to the literature (cf. [3]) and confine ourselves to formalities essential to our contribution: Given an alphabet \mathcal{P} , a *logic program* is a finite set of rules, r , of the form $p_0 \leftarrow p_1, \dots, p_m, \text{not } p_{m+1}, \dots, \text{not } p_n$ where $n \geq m \geq 0$ and $p_i \in \mathcal{P}$ is an *atom* for $0 \leq i \leq n$. Let $\text{head}(r) = p_0$ be the *head* of r and $\text{body}(r) = \{p_1, \dots, p_m, \text{not } p_{m+1}, \dots, \text{not } p_n\}$ be the *body* of r . The set of atoms occurring in a logic program Π is given by $\text{atom}(\Pi)$. The set of bodies in Π is $\text{body}(\Pi) = \{\text{body}(r) \mid r \in \Pi\}$. An *answer set* of a logic program Π is a model of Π that satisfies a certain “stability criterion” (cf. [3]).

We consider the following systems: (i) *smodels* [7], as an atom-based ASP-solver, assigning truth values to atoms only; (ii) *dlv* [5], also atom-based, yet designed for handling disjunctive logic programs; (iii) *nomore++* [1], pursuing a hybrid approach, wherein truth values are assigned to both atoms and bodies; (iv) a restriction of *nomore++* to assignments to bodies only, indicated by *nomore++^B*.⁴ All systems incorporate techniques such as propagation and lookahead in their solving procedures [1, 7].

Inspired by structural normal form translations [2], we give two simple transformations introducing new symbols for bodies and atoms, respectively. This allows standard ASP-solvers to access the underlying structures and implicitly incorporate them in their solving procedures. To this end, we extend the alphabet \mathcal{P} of a program Π by adding a new atom α_a for each $a \in \text{atom}(\Pi)$ and a new atom β_B for each $B \in \text{body}(\Pi)$. We then define a transformation over the extended alphabet:

$$\mathcal{T}_B(\Pi) = \left\{ \begin{array}{lcl} \beta_B & \leftarrow & B \\ \text{head}(r) & \leftarrow & \beta_B \end{array} \mid r \in \Pi, B = \text{body}(r) \right\}$$

This enables ASP-solvers branching on atoms, like *smodels*, to indirectly incorporate bodies as computable objects in their solving procedures. Vice versa, a different transformation over the extended

¹ Universität Potsdam, Postfach 90 03 27, D–14439 Potsdam, Germany

² Helsinki University of Technology, P.O. Box 5400, FI-02015 TKK, Finland.

³ Affiliated with Simon Fraser University, Canada.

⁴ To distinguish full-fledged *nomore++* in (iii) from this restricted variant, we sometimes add superscript $a + B$, yielding *nomore++^{a+B}*.

alphabet enables body-based systems (such as *nomore++^B*) to indirectly incorporate atoms:

$$\mathcal{T}_a(\Pi) = \left\{ \begin{array}{lcl} \alpha_a & \leftarrow & \text{body}(r) \\ a & \leftarrow & \alpha_a \end{array} \mid r \in \Pi, a = \text{head}(r) \right\}$$

As an example, consider program Π along with its transforms:

$$\begin{aligned} \Pi &= \left\{ \begin{array}{lcl} a & \leftarrow & b, \text{not } c \\ a & \leftarrow & d, \text{not } e \end{array} \right\} \\ \mathcal{T}_B(\Pi) &= \left\{ \begin{array}{lcl} \beta_{\{b, \text{not } c\}} & \leftarrow & b, \text{not } c \\ a & \leftarrow & \beta_{\{b, \text{not } c\}} \\ \beta_{\{d, \text{not } e\}} & \leftarrow & d, \text{not } e \\ a & \leftarrow & \beta_{\{d, \text{not } e\}} \end{array} \right\} \\ \mathcal{T}_a(\Pi) &= \left\{ \begin{array}{lcl} \alpha_a & \leftarrow & b, \text{not } c \\ \alpha_a & \leftarrow & d, \text{not } e \\ a & \leftarrow & \alpha_a \end{array} \right\} \end{aligned}$$

To begin with, we empirically underpin our claim that treating atoms and bodies as equitable computational objects may yield exponential speed-ups by some experimental results. As common in proof complexity [4], we need an infinite family of logic programs witnessing an exponential behavior in the best-case. For this, we consider the following programs Π_B^n and Π_a^n for $n \geq 0$:

$$\begin{aligned} \Pi_B^n &= \left\{ \begin{array}{l} r_0 : x \leftarrow \text{not } x \\ r_1 : x \leftarrow \text{not } a_1, \text{not } b_1 \\ \vdots \\ r_n : x \leftarrow \text{not } a_n, \text{not } b_n \\ r_{n+1} : a_1 \leftarrow \text{not } b_1 & r_{n+2} : b_1 \leftarrow \text{not } a_1 \\ \vdots \\ r_{3n-1} : a_n \leftarrow \text{not } b_n & r_{3n} : b_n \leftarrow \text{not } a_n \end{array} \right\} \\ \Pi_a^n &= \left\{ \begin{array}{ll} r_0 : x \leftarrow c_1, \dots, c_n, \text{not } x & \\ r_1 : c_1 \leftarrow \text{not } a_1 & r_2 : c_1 \leftarrow \text{not } b_1 \\ \vdots & \vdots \\ r_{2n-1} : c_n \leftarrow \text{not } a_n & r_{2n} : c_n \leftarrow \text{not } b_n \\ r_{2n+1} : a_1 \leftarrow \text{not } b_1 & r_{2n+2} : b_1 \leftarrow \text{not } a_1 \\ \vdots & \vdots \\ r_{4n-1} : a_n \leftarrow \text{not } b_n & r_{4n} : b_n \leftarrow \text{not } a_n \end{array} \right\} \end{aligned}$$

None of these programs has an answer set, that is, all of them are *unsatisfiable*. Importantly, this can be determined *linearly*. In both programs, the source of unsatisfiability lies in r_0 . For Programs Π_B^n , this can be found out linearly by assigning “true” to bodies of the form $\{\text{not } a_i, \text{not } b_i\}$ ($1 \leq i \leq n$) and by subsequent propagation. For *smodels*, transformation \mathcal{T}_B makes these bodies indirectly accessible. With Programs Π_a^n , assigning “false” to an atom c_i ($1 \leq i \leq n$) leads to a conflict by propagation. For *nomore++^B*, transformation \mathcal{T}_a makes these atoms indirectly accessible. The programs are composed in such a way that lookahead on the crucial structures (bodies and atoms, respectively) is able to de-

termine the optimal choices and thus ensures that the minimal number of choices is obtained.⁵ Unlike this, detecting the unsatisfiability of Π_B^n with the atom-based *smodels* strategy requires exponentially many choices. The same holds for Π_a^n with *nomore++^B*'s strategy. The *dlv* system employs a slightly different lookahead strategy with fewer lookahead operations, which is why it is unable to identify the best choices for the given program classes. We give *d lv* results purely for the record.

We have implemented the above transformations and run benchmarks on a number of instances of Π_B^n and Π_a^n and their transforms. The implementation is available at [6]. Results of these benchmarks are given in Table 1 and 2. All tests were run on an AMD Athlon 1.4GHz PC with 512MB RAM. We limited memory to 256MB and time to 900s. All results are given in terms of number of choices⁶ and seconds (in parentheses), reflecting the average of 10 runs. The application of a translation is indicated by T_B or T_a , respectively, otherwise a hyphen indicates no such application. We see that the application of transformation T_B leads to an exponential speed-up for *smodels* on Π_B^n , while T_a exponentially speeds up *nomore++^B* on Π_a^n . On the other hand, a superfluous transformation (as in the case of $T_a(\Pi_a^n)$ when running *smodels*) seems not to hurt the performance significantly. In terms of choices, the hybrid approach of *nomore++^{a+B}* lets it perform optimally on both Π_B^n and Π_a^n .

n	<i>d lv</i> (2006.01.12)		<i>smodels</i> (2.28)		<i>nomore++ (1.4)</i>			
	-	T_B	-	T_B	-	T_B	-	T_B
2	(0.00)	(0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
4	(0.00)	(0.01)	3 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
6	(0.00)	(0.01)	15 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.02)
8	(0.00)	(0.01)	63 (0.01)	0 (0.01)	0 (0.02)	0 (0.02)	0 (0.01)	0 (0.02)
10	(0.01)	(0.03)	255 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
12	(0.05)	(0.07)	1,023 (0.02)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
14	(0.20)	(0.24)	4,095 (0.03)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
16	(0.81)	(0.93)	16,383 (0.12)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
18	(3.25)	(3.76)	65,535 (0.45)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
20	(13.16)	(15.05)	262,143 (1.77)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
22	(53.06)	(61.17)	1,048,575 (7.09)	0 (0.02)	0 (0.01)	0 (0.03)	0 (0.01)	0 (0.02)
24	(213.93)	(247.57)	4,194,303 (28.54)	0 (0.02)	0 (0.01)	0 (0.03)	0 (0.01)	0 (0.02)
26	(862.69)	> 900	16,777,215 (114.56)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
28	> 900	> 900	67,108,863 (460.30)	0 (0.02)	0 (0.02)	0 (0.03)	0 (0.02)	0 (0.03)
30	> 900	> 900	> 900	0 (0.03)	0 (0.02)	0 (0.03)	0 (0.02)	0 (0.03)

Table 1. Results for family $\{\Pi_B^n\}$.

n	<i>d lv</i> (2006.01.12)		<i>smodels</i> (2.28)		<i>nomore++ (1.4)</i>			
	-	T_a	-	T_a	-	T_a	-	T_a
2	(0.00)	(0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
4	(0.00)	(0.01)	0 (0.01)	0 (0.01)	3 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
6	(0.00)	(0.01)	0 (0.01)	0 (0.01)	15 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
8	(0.01)	(0.02)	0 (0.01)	0 (0.02)	63 (0.02)	0 (0.02)	0 (0.01)	0 (0.02)
10	(0.02)	(0.05)	0 (0.01)	0 (0.02)	255 (0.03)	0 (0.02)	0 (0.01)	0 (0.02)
12	(0.06)	(0.16)	0 (0.01)	0 (0.01)	1,023 (0.07)	0 (0.02)	0 (0.01)	0 (0.02)
14	(0.24)	(0.60)	0 (0.01)	0 (0.02)	4,095 (0.25)	0 (0.02)	0 (0.01)	0 (0.02)
16	(0.95)	(2.47)	0 (0.01)	0 (0.02)	16,383 (0.98)	0 (0.02)	0 (0.01)	0 (0.02)
18	(3.86)	(10.18)	0 (0.01)	0 (0.02)	65,535 (3.88)	0 (0.03)	0 (0.01)	0 (0.02)
20	(15.61)	(42.04)	0 (0.01)	0 (0.02)	262,143 (15.51)	0 (0.03)	0 (0.02)	0 (0.02)
22	(63.36)	(173.98)	0 (0.01)	0 (0.02)	1,048,575 (61.99)	0 (0.03)	0 (0.02)	0 (0.02)
24	(257.14)	(718.17)	0 (0.01)	0 (0.02)	4,194,303 (248.74)	0 (0.03)	0 (0.02)	0 (0.02)
26	> 900	> 900	0 (0.02)	0 (0.02)	> 900	0 (0.03)	0 (0.02)	0 (0.03)
28	> 900	> 900	0 (0.01)	0 (0.03)	> 900	0 (0.04)	0 (0.02)	0 (0.03)
30	> 900	> 900	0 (0.02)	0 (0.03)	> 900	0 (0.04)	0 (0.02)	0 (0.03)

Table 2. Results for family $\{\Pi_a^n\}$.

Finally, let us show that our observation is not accidental and formally prove that the availability of both atoms and bodies may lead to exponentially smaller search spaces than obtainable in either restricted case. To this end, we apply well-known concepts from *proof complexity* [4] and show that there are infinite witnessing families of programs for which even the best-case complexity is exponential. To

⁵ With lookahead disabled, propagation does not suffice to detect unsatisfiability. Thus, linearly many choices must be made.

⁶ The choices of *d lv* are omitted since they rely on a different concept.

be more precise, we show that minimal refutations for Π_B^n and Π_a^n , i.e. proofs that the respective programs have no answer sets, require exponentially many choices in the cases of Π_B^n with *smodels* and Π_a^n with *nomore++^B*, while a linear number of choices is required for the transformed programs.

Theorem 1 *There is an infinite family $\{\Pi^n\}$ of logic programs such that the minimal number of choices made by *smodels* for determining the unsatisfiability of Π^n is $O(2^n)$ and of $T_B(\Pi^n)$ is $O(n)$.*

Theorem 2 *There is an infinite family $\{\Pi^n\}$ of logic programs such that the minimal number of choices made by *nomore++^B* for determining the unsatisfiability of Π^n is $O(2^n)$ and of $T_a(\Pi^n)$ is $O(n)$.*

In neither case is it possible to simulate this behavior polynomially, as witnessed by families $\{\Pi_B^n\}$ and $\{\Pi_a^n\}$, respectively. Finally, we note that the hybrid approach pursued by *nomore++^{a+B}* gives a best-case complexity of $O(n)$ for both classes of programs: A hybrid approach can polynomially simulate both uniform approaches.

2 DISCUSSION

We have shown that the integration of bodies as explicitly referable objects into atom-based ASP-solvers may have a great computational impact. The same holds for purely body-based (or rule-based) approaches, when introducing special atoms. Considering bodies rather than more complex formulas is motivated by the fact that they allow for replacing rules in characterizing computational concepts (see below) and by applying an “input strategy” (similar to *linear resolution*). Very simple transformations allow for exponential reductions of the best-case complexity on witnessing families of logic programs. That is, any uniform ASP-solver even equipped with the optimal heuristics will be unable to solve all witnessing programs in polynomial time. This is only possible by means of a genuinely hybrid approach, as realized in *nomore++*, or via respective transformations.

Our discovery is insofar surprising as standard ASP-solvers, such as *smodels*, rely already on twofold data structures but somehow stop “halfway”: The abstract *smodels* algorithm [7] relies on atoms (in branching and assigning truth values). However, underlying propagation and its implementation must also take rules (or bodies) into account. Consequently, the primary data structure of *smodels* is an “atom-rule graph.” Furthermore, propagation takes advantage of the concept of “active” rules, that is, rules whose bodies are not false [7]. Although bodies are vital objects in *smodels*’ propagation, they are ignored when branching.

Acknowledgements. The first, second, and fourth author were supported by DFG under grant SCHA 550/6-4, TP C.

REFERENCES

- [1] C. Anger, M. Gebser, T. Linke, A. Neumann, and T. Schaub, ‘The *nomore++* approach to answer set solving’, in *Proceedings of LPAR'05*, eds., G. Sutcliffe and A. Voronkov, pp. 95–109. Springer-Verlag, (2005).
- [2] M. Baaz, U. Egly, and A. Leitsch, ‘Normal form transformations’, in *Handbook of Automated Reasoning*, eds., J. Robinson and A. Voronkov, 273–333, Elsevier, (2001).
- [3] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.
- [4] P. Beame and T. Pitassi, ‘Propositional proof complexity: Past, present, and future’, *Bulletin of EATCS*, **65**, 66–89, (1998).
- [5] N. Leone, W. Faber, G. Pfeifer, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri, and F. Scarcello, ‘The DLV system for knowledge representation and reasoning’, *ACM TOCL*, (2006). To appear.
- [6] <http://www.cs.uni-potsdam.de/nomore>.
- [7] P. Simons, ‘Extending and implementing the stable model semantics’, Helsinki University of Technology, (2000). Doctoral dissertation.

Irrelevant Updates of Nonmonotonic Knowledge Bases

Ján Šefránek and Jozef Šiška¹

1 INTRODUCTION

The second postulate of Katsuno and Mendelzon, 2[KM] hereafter, characterizes irrelevant updates: if ψ implies μ , then $\psi \diamond \mu$ is equivalent to ψ , where ψ is a knowledge base, μ is an update and \diamond is an update operator [3].

We show that the postulate has to be modified, if nonmonotonic assumptions are considered. Our characterization of irrelevant updates is based on a dependency framework [5], which provides an alternative semantics of multidimensional dynamic logic programming (MDyLP). MDyLP [4] can be viewed as a logical idealization of nonmonotonic knowledge bases (NMKB).

Unwanted generation of new models caused by cyclic or tautological updates has been a serious problem of MDyLP for a time. Recently, the problem has been solved for dynamic logic programs in [1] and for the general MDyLP in [2]. The solution of [1] is based on Refined Extension Principle with ambition to express features essential for a “right” semantics of logic program updates. Unfortunately, the principle has not been extended to the general case of MDyLP and the trivial semantics assigning empty set of models to each dynamic logic program satisfies the principle, see [2].

We believe that the role of nonmonotonic assumptions (and dependencies on assumptions) is crucial for understanding the semantic problems of NMKBs and also for understanding unwanted generation of models.

2 DEPENDENCY FRAMEWORK

We assume a language \mathcal{L} with a set of atoms \mathcal{A} . In general, a language of the dependency framework contains a set of assumptions. In this extended abstract are assumptions represented by a set of negative literals $\mathcal{D} = \{\text{not } A \mid A \in \mathcal{A}\}$, where *not* is a default negation. The set of literals is $\text{Lit} = \mathcal{A} \cup \mathcal{D}$.

Definition 1 A dependency relation is a set of pairs $\{(L, W) \mid L \in \text{Lit}, W \subseteq \text{Lit}, L \notin W\}$.

$Xs \subseteq \mathcal{D}$ is called a sound set of assumptions (SSOA) w.r.t. the dependency relation \ll iff the set $Cn_{\ll}(Xs) = \{L \in \text{Lit} \mid L \ll Xs\} \cup Xs$ is non-empty and consistent.

It is said that a SSOA Xs is total (TSSOA) iff for each $A \in \mathcal{A}$ holds either $A \in Cn_{\ll}(Xs)$ or $\text{not } A \in Cn_{\ll}(Xs)$. The set of all (T)SSOAs w.r.t. \ll is denoted by $(T)SSOA(\ll)$. \square

Each NMKB consisting of rules can be mapped into the dependency framework. We will present a mapping of generalized logic programs (programs hereafter) and multidimensional dynamic logic programs (multiprograms hereafter) into the dependency framework.

A program consists of a set of rules of the form $L_0 \leftarrow L_1, \dots, L_k$, where L_i is a literal. L_0 is denoted by $\text{head}(r)$ and $\{L_1, \dots, L_k\}$ by $\text{body}(r)$, if a rule r is considered. A multiprogram is a set of programs together with a preference relation – a partial order on the set of programs. If there are some conflicts between programs, they are solved according to the preference relation – information from the less preferred programs is rejected.

Definition 2 A literal L depends on a set of literals W , $L \notin W$, with respect to a program P ($L \ll_P W$) iff there is a sequence of rules $\langle r_1, \dots, r_k \rangle$, $k \geq 1$, and $\text{head}(r_k) = L$, $W \models \text{body}(r_1)$, for each i , $1 \leq i < k$, $W \cup \{\text{head}(r_1), \dots, \text{head}(r_i)\} \models \text{body}(r_{i+1})$.

It is said that the dependency relation \ll_P is generated by the program P . \square

Notice that a literal cannot depend on itself (also in a context of other literals). Next theorem shows that a semantics of the dependency framework based on TSSOAs is equivalent to stable model (answer set) semantics of logic programs.

Theorem 3 X is a TSSOA w.r.t. \ll_P iff $Cn_{\ll_P}(X)$ is a stable model of P . Let S be a stable model of P . Then there is $X \subseteq \mathcal{D}$, a TSSOA w.r.t. \ll_P such that $S = Cn_{\ll_P}(X)$. \square

Next example illustrates Theorem 3 and it also shows that dependencies in multiprograms are well defined. Only the most simple multiprograms of the form $\langle P, U \rangle$, where U is more preferred (updating) program are considered here because of the limited space. However, the results are extendable to the general case.

Example 4 $P = \{a \leftarrow \text{not } b; b \leftarrow \text{not } a\}$, $U = \{\text{not } b \leftarrow \text{not } a\}$.

There are two TSSOAs w.r.t. \ll_P : $Xs_1 = \{\text{not } b\}$, $Xs_2 = \{\text{not } a\}$. $Cn_{\ll_P}(Xs_1) = \{\text{not } b, a\}$, $Cn_{\ll_P}(Xs_2) = \{\text{not } a, b\}$. Note that both TSSOAs generate (all) stable models of P .

$P \cup U$ is a program and it generates a dependency relation. Observe that $a \ll_{P \cup U} \{\text{not } a\}$, but $(a, \{\text{not } a\}) \notin (\ll_P \cup \ll_U)$. \square

Proposition 5 Let $\langle P, U \rangle$ be a multiprogram. Then $\ll_{P \cup U}$ is well defined. It holds $(\ll_P \cup \ll_U) \subseteq \ll_{P \cup U}$, but the converse inclusion does not hold.

Multiprograms contain in general conflicts. Example 6 illustrates notions connected to conflicts and solutions of conflicts.

Example 6 Dependency relation $\ll_{P \cup U}$ from Example 4 contains conflicts $C_1 = \{(b, \{\text{not } a\}), (\text{not } b, \{\text{not } a\})\}$ and $C_2 = \{(a, \{\text{not } a\})\}$.

A solution of a conflict is a set of dependencies which should be rejected (ignored) in order to remove the conflict. $D =$

¹ Comenius University, Bratislava, Slovakia, email: [sefranek,siska]@fmph.uniba.sk

$\{(not\ b, \{not\ a\})\}$ is a minimal solution of C_1 . However, it is more suitable to reject less preferred dependencies. Hence, $D' = \{(b, \{not\ a\})\}$ is a more suitable solution of C_1 than D . A solution of a conflict is called a good solution of it if there is not a more suitable solution. D' is the good solution of C_1 .

We intend to define the semantics of a multiprogram (with the dependency relation \ll) as a subset of \ll , which provides a coherent view on the multiprogram – i.e. there is a TSSOA w.r.t. the subset. In our example the set of assumptions $\{not\ a, not\ b\}$ is a TSSOA w.r.t. $View = \ll_{P \cup U} \setminus \{(b, not\ a)\}$ and $\{not\ b\}$ is a TSSOA w.r.t. $\ll_{P \cup U}$. \square

Definition 7 (Semantics of multiprograms) A dependency relation \ll is called *coherent* iff there is a TSSOA w.r.t. \ll .

Semantics of multiprograms (of the form $\langle P, U \rangle$) is a mapping Σ which assigns to $\langle P, U \rangle$ the set of all pairs of the form $(Z, View)$, where $View$ is a coherent subset of $\ll_{P \cup U}$ and Z is a TSSOA w.r.t. $View$.

3 IRRELEVANT UPDATES

2[KM] is adapted for multiprograms of the form $\langle P, U \rangle$ as follows: if $P \models_{SM} U$, then $TSSOA(P \cup U) = TSSOA(P)$, where $P \models_{SM} U$ means that U is satisfied in all stable models of P . Unfortunately, 2[KM] is not acceptable literally for multiprograms (and for NMKBs), see Examples 10 and 11. 2[KM] should be modified for NMKBs in order to catch the role of nonmonotonic assumptions.

Convention 8 Let $\langle P, U \rangle$ be a multiprogram. We would call U an *irrelevant update* of P iff $TSSOA(\ll_{P \cup U}) = TSSOA(\ll_P)$. \square

We are aiming to find sufficient and necessary conditions of $TSSOA(\ll_{P \cup U}) = TSSOA(\ll_P)$ and to define irrelevant updates in terms of those conditions.

First, consider the condition $P \models_{SM} U$. It is a necessary condition of an irrelevant update. On the other hand, if $P \not\models_{SM} U$, then update U is a relevant one.

Proposition 9 If $P \not\models_{SM} U$, then $TSSOA(\ll_P) \not\subseteq TSSOA(\ll_{P \cup U})$. If $TSSOA(\ll_{P \cup U}) = TSSOA(\ll_P)$, then $P \models_{SM} U$. If $P \models_{SM} U$, then $TSSOA(\ll_P) \subseteq TSSOA(\ll_{P \cup U})$.

Unfortunately, Examples 10 and 11 show that $TSSOA(\ll_{P \cup U}) \subseteq TSSOA(\ll_P)$ does not hold in general, if $P \models_{SM} U$.

Example 10

$$P = \{a \leftarrow not\ b\} \quad U = \{b \leftarrow not\ a\}$$

$P \models_{SM} U$, but U introduces a new assumption $not\ a$, which is false in all stable models of P and generates a new stable model of $P \cup U$.

Example 11 $P = \{a_1 \leftarrow not\ b_1; b_1 \leftarrow not\ a_1; a_2 \leftarrow not\ b_2; not\ b_2 \leftarrow not\ a_2\}$, $U = \{b_2 \leftarrow not\ a_2\}$.

The set of assumptions $\{not\ a_2\}$ is a SSOA w.r.t. \ll_U and $Cn_{\ll_P}(\{not\ a\}) \cup Cn_{\ll_U}(\{not\ a\})$ is inconsistent. The inconsistency can be overridden if we prefer $b_2 \ll_U \{not\ a_2\}$ over $not\ b_2 \ll_P \{not\ a_2\}$. Hence, $\{not\ a_2\}$ generates new stable models: $\{not\ a_2, not\ b_1\}$ and $\{not\ a_2, not\ a_1\}$ are TSSOAs w.r.t. $\ll_{P \cup U} \setminus \{(not\ b_2, not\ a_2)\}$. \square

We proceed to the definition of irrelevant updates. An update U of P is irrelevant if $P \models_{SM} U$ and there is no set of assumptions, which is sound w.r.t. \ll_U , some literals are dependent on it, it is not falsified, not satisfied in P (in order to gain something new w.r.t. $P \models_{SM} U$), and which generates a new model (by extending the set of reasonable assumptions, see Example 10, or by solving a conflict, see Example 11).

Definition 12 An assumption $not\ A$, where $A \in \mathcal{A}$, is *falsified* in a dependency relation \ll iff $A \ll \emptyset$, $not\ A \not\ll \emptyset$ and \emptyset is a SSOA w.r.t. \ll .

A set of assumptions $Xs \subseteq \mathcal{D}$ is falsified in \ll iff it contains a literal falsified in \ll . \square

Definition 13 (Irrelevant update) Let $\langle P, U \rangle$ be a multiprogram, P be coherent and $P \models_{SM} U$.

It is said that U is an irrelevant update of P iff there is no $Xs \subseteq \mathcal{D}$ such that:

- $Xs \in SSOA(\ll_U)$ and $Cn_{\ll_U}(Xs) \setminus Xs \neq \emptyset$,
- Xs is not falsified in $\ll_{P \cup U}$,
- Xs is false in each stable model of P ,
- there is $Bss \subseteq \mathcal{D}$ s.t. $Xs \subseteq Bss$ and $Bss \in TSSOA(\ll_{P \cup U})$ or
 - $Cn_{\ll_P}(Xs) \cup Cn_{\ll_U}(Xs)$ is inconsistent,
 - there is $View \subseteq \ll_{P \cup U}$ and $Bss \subseteq \mathcal{D}$ such that $Xs \subseteq Bss$ and $Bss \in TSSOA(View)$ \square

Observe that the condition $Cn_{\ll_U}(Xs) \setminus Xs \neq \emptyset$ eliminates unsupported cyclic updates. A nondeterministic algorithm for computation of TSSOAs is presented in [5].

Theorem 14 If U is an irrelevant update of P , then

$$TSSOA(\ll_{P \cup U}) = TSSOA(\ll_P).$$

4 CONCLUSIONS

Main contributions of this work are as follows:

- analysis and definition of irrelevant updates,
- an adaptation of 2[KM] for updates of NMKB,
- we show that irrelevant updates do not generate new models and that models of the original program are preserved after an irrelevant update.

ACKNOWLEDGMENTS

This work was supported under grants APVV-20-P04805 and VEGA 1/3112/06. We are grateful to one of the anonymous reviewers for his comments.

REFERENCES

- [1] Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. *Studia Logica* **1** (2005)
- [2] Banti, F., Alferes, J.J., Brogi, A., Hitzler, P.: The well supported semantics for multidimensional dynamic logic programs. LPNMR 2005, LNCS 3662, Springer, 356-368
- [3] Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. Proc. of KR '91
- [4] Leite, J.A., Alferes, J.J., Pereira, L.M.: Multi-dimensional dynamic logic programming. In: Procs. of CLIMA'00. (2000) 17–26
- [5] Šefránek, J.: Rethinking semantics of dynamic logic programming. To appear in Proc. of NMR 2006

Decision making in large-scale domains: a case study¹

Mikhail Soutchanski and Huy Pham² and John Mylopoulos³

Abstract.

Planning under uncertainty attracted significant attention in AI and in other fields. To overcome computational problems associated with Markov Decision Processes (MDPs) in large scale domains researchers often take advantage of structural properties and look for approximately optimal solutions. DTGolog, a decision-theoretic agent programming language based on the situation calculus, was proposed to ease some of the computational difficulties by using natural ordering constraints on execution of actions. Using DTGolog, domain specific constraints on the set of available policies can be expressed in a high-level program and this program helps to reduce significantly computation required to find a policy optimal in this set. Our paper explores whether the DTGolog framework can be used to evaluate different designs of a decision making agent in a large real-world domain. Each design is understood as combination of a template (expressed as a Golog program) for available policies and a reward function. To evaluate and compare alternative designs we estimate the probability of goal satisfaction for each design. As a domain, we choose the London Ambulance Service (LAS) case study that is well known in software engineering, but remains unknown in AI. In our paper we demonstrate that DTGolog can be applied successfully to quantitative evaluation of alternative designs in terms of their ability to satisfy a system goal with a high probability. We provide a detailed axiomatization of the domain in the temporal situation calculus with stochastic actions. The main advantage of this representation is that neither actions, nor states require explicit enumeration. We do an experimental analysis using an on-line implementation of DTGolog coupled with a simulator that models real time actions of many external agents.

1 Introduction and Motivation

There is a significant amount of work done in AI related to planning under uncertainty when a certain high level goal must be satisfied with a high probability [7, 1, 10, 9, 16]. Most of the proposed approaches are related to solving a decision-theoretic planning problem in relatively small domains (with less than 10^9 states). However, there are many practical domains where the task of designing a decision making agent (that guarantees goal satisfaction with a sufficiently high probability) is difficult due to a very large number of the state features and (ground) actions with uncertain effects. In these domains, the main problem is that state of the art planners cannot scale up to compute (or approximate) an optimal policy due to extremely

large size of the state space. Even the task of computing the value of a single policy can be prohibitively difficult in these domains. The second common problem is that in some domains the goal of interest is characterized in terms of quality of an on-going process driven by external agents. To deal with the first problem (scalability), one can try to use a logical representation that avoids explicit state and action enumeration. In addition, one can try to elaborate alternative designs of a decision making agent by goal-means analysis, e.g., using goal regression, a mechanism well studied in AI [15, 12]. By careful refining a goal that must be (at least partially) satisfied into sub-goals, and then by identifying sub-tasks to solve and primitive actions that must be executed to solve these sub-tasks, it is possible to ease to some degree the computational burden of designing such an agent. Indeed, a gradual refinement process can identify useful sequences, loops, conditional or recursive structures of actions that provide together important constraints on the set of policies that need to be considered, and as a consequence, significantly reduce the number of potential policies that ever need to be analyzed. One can imagine also that such analysis can identify where search between alternative actions must concentrate: this can be indicated by nondeterministic choices between alternatives. In realistic domains, this refinement process can lead to different designs depending on how stakeholder goals will be captured in this process. Because this can lead to a variety of designs that need precise evaluation, the problem of designing a decision making agent can be reduced to quantitative evaluation of those different designs of an agent which have been elaborated during the goals-means refinement process. To deal with the second problem (representation of an ongoing interaction with external agents), one can build a simulator of exogenous actions and evaluate all identified alternative designs with respect to the same simulator (see [11] for details).

We discuss an expressive framework that provides a tool for reasoning about different designs. This framework, a decision-theoretic extension of Golog (DTGolog), was first introduced in the context of designing efficient controllers for mobile robots [2, 13]. Later, it was extended to the multi-person games [4], was successfully adapted to program robots playing soccer [3], and was also extended to incorporate qualitative preferences to personalize Web services [5].

To the best of our knowledge, there were no attempts to use the DTGolog framework as a tool for evaluation of alternative designs of a decision making agent functioning in a large-scale domain characterized by on-going interaction with many external agents. All previous approaches applied DTGolog to the task of computing a policy in a finite horizon decision-theoretic planning problem. We would like to fill this gap and do extensive analysis of DTGolog with an application to a real case study. In particular, we explore a well-known case study (LAS-CAD) that received a significant attention in the software engineering literature, but remains unknown to the AI community [14, 6, 8]. It is an excellent example of a problem with probabilistic goals. This case study comes from an investigation into a failed software development project. We suggest this case study as a grand challenge for research on planning under uncertainty.

¹ A full version of this paper is available as <http://www.cs.toronto.edu/~mes/papers/LAS/long.pdf>

² Department of Computer Science, Ryerson University, 245 Church Street, ENG281, Toronto, Ontario, M5B 2K3, Canada email: {mes, hpham}@scs.ryerson.ca

³ Department of Computer Science, University of Toronto, Toronto, Ontario, M5S 3G4, Canada email: jm@cs.toronto.edu

We try to keep our presentation as generic as possible to indicate how our modeling framework can be applied or extended to other decision-making environments. In particular, we clearly show the main features of DTGolog model that can be applied to other domains as well: actions, fluents, precondition axioms, successor-state axioms, initial database, transition probabilities, rewards, Golog procedures for expressing natural constraints on decision making. We illustrate all these main features of the specification framework using the case study.

The main contributions of our paper are the following. First, we developed an extensive logical formalization of a non-trivial domain. Second, we demonstrated that DTGolog is well suited to the task of evaluation of alternative designs of a decision making agent. Third, we did experimental analysis of three different designs of a decision making agent using the same simulator for a fair comparison. Each design is understood as a combination of a Golog program and a reward function. We use two different reward functions to show that preferences of stakeholders can be captured differently in the model. We also intentionally use two Golog programs to show that different decision making designs can be represented using this approach.

The interested reader can consult [12] for background on the temporal situation calculus, stochastic actions and Golog, [1] for background on MDPs, and [13] and papers mentioned above for background on DTGolog. The full version of this paper also includes background material necessary to understand this paper.

2 Discussion and Conclusion

We considered applicability of DTGolog to the task of evaluation of alternative designs of a decision making agent. We choose a domain where the state space has well beyond $30^{300} \cdot 2^{300}$ states (3 grid-worlds 10×10 with at least 1 request anywhere and 30 cars located anywhere). Consequently, it is doubtful that even state-of-the art MDP solvers (such as SPUDD) can solve the decision-theoretic planning problem in this domain and can be used for comparison with DTGolog. Moreover, our task was not solving the decision-theoretic planning problem (exactly or approximately), but evaluating alternative designs. DTGolog was previously used only to compute (approximately optimal) policies for finite horizon planning problems, but we consider a process-oriented problem that continues indefinitely (as long as new exogenous requests arrive). In this setting, we are interested whether DTGolog is a tool that can be used for quantitative evaluation of alternative designs of a resource allocating agent, where each design is determined by a Golog program and a reward function. We demonstrate that alternative designs can be compared by using a simulator in addition to an on-line DTGolog interpreter. We show that domain dependent constraints on the set of policies can range from a purely deterministic program (no decision making at all) to a Golog program with a limited number of decision points. In the latter case, a finite horizon planning can be accomplished by the off-line DTGolog interpreter. It might be surprising that some of our designs are non-deterministic Golog program where non-determinism is resolved at run time to make the best decision (with respect to program constraints). However, we believe this is important, because resolving this non-determinism at design time can be problematic due to the huge size of the state space.

The large number of choices and the large number of actions with uncertain outcomes present computational challenges that have to be addressed in future work. The most important direction for future research is overcoming computational challenges of the DTGolog framework: using sampling to deal with large branching factor (in

the version of directed value iteration that provides semantics for a DTGolog interpreter [2]) and using progression to deal with long situations [12]. Our research goal is a more advanced framework to handle models that are large enough to be of use in software design applications such as this one. In 2004, the real LAS-CAD system included about 30 regions, about 400 vehicles and was the largest public ambulance system in the world.⁴

REFERENCES

- [1] Craig Boutilier, Thomas Dean, and Steve Hanks, ‘Decision theoretic planning: Structural assumptions and computational leverage’, *J. of Artificial Intelligence Research*, **11**, 1–94, (1999).
- [2] Craig Boutilier, Ray Reiter, Mikhail Soutchanski, and Sebastian Thrun, ‘Decision-theoretic, high-level agent programming in the situation calculus’, in *Proceedings of the 17th Conference on Artificial Intelligence (AAAI-00)*, pp. 355–362, Austin, TX, (July 30–3 2000). AAAI Press.
- [3] A. Ferrein, Ch. Fritz, and G. Lakemeyer, ‘Using Golog for deliberation and team coordination in robotic soccer’, *KI Künstliche Intelligenz*, **05**(1), 24–43, (2005).
- [4] Alberto Finzi and Thomas Lukasiewicz, ‘Game-theoretic agent programming in Golog’, in *Proceedings of the 16th biennial European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, (August 2004). IOS.
- [5] Christian Fritz and Sheila McIlraith, ‘Compiling qualitative preferences into decision-theoretic Golog programs’, in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*, ed., John Mylopoulos, Lake District, UK, (June 2006).
- [6] Jeff Kramer and Alexander Wolf, ‘Proceedings of the 8th international workshop on software specification and design’, *ACM SIGSOFT Software Engineering Notes*, **21**(5), 21–35, (September 1996).
- [7] Nicholas Kushmerick, Steve Hanks, and Daniel S. Weld, ‘An algorithm for probabilistic planning’, *Artificial Intelligence, Special Issue on Planning and Scheduling*, **76**(1–2), 239–286, (July 1995).
- [8] Emmanuel Letier and Axel van Lamsweerde, ‘Reasoning about partial goal satisfaction for requirements and design engineering’, in *Proceedings of the 12th International Symposium on the Foundation of Software Engineering FSE-04*, pp. 53–62, Newport Beach, CA, (November 2004). ACM Press.
- [9] Omid Madani, Steve Hanks, and Anne Condon, ‘On the undecidability of probabilistic planning and related stochastic optimization problems’, *Artificial Intelligence, Special Issue on Planning with Uncertainty and Incomplete Information*, **147**(1–2), 5–34, (July 2003).
- [10] S.M. Majercik and M.L. Littman, ‘Contingent planning under uncertainty via stochastic satisfiability’, *Artificial Intelligence*, **147**(1–2), 119–162, (July 2003).
- [11] Huy Pham, Mikhail Soutchanski, and John Mylopoulos, ‘A simulator and a Golog implementation of the London Ambulance Service (LAS) Computer-Aided Dispatch (CAD) system’, Technical report, Department of Computer Science, Ryerson University, <http://www.cs.toronto.edu/~mes/papers/LAS/index.html>, (2006).
- [12] Raymond Reiter, *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*, MIT Press, 2001.
- [13] Mikhail Soutchanski, *High-Level Robot Programming in Dynamic and Incompletely Known Environments*, Ph.D. Thesis, Computer Science Dep., University of Toronto, <http://www.cs.toronto.edu/~mes/papers/index.html>, 2003.
- [14] The Communications Directorate, *Report of the Inquiry Into The London Ambulance Service (South West Thames Regional Health Authority)*, The 8th International Workshop on Software Specification and Design Case Study. Electronic Version prepared by Anthony Finkelstein. Available at <http://www.cs.ucl.ac.uk/staff/A.Finkelstein/las.html>, February 1993.
- [15] Richard Waldinger, ‘Achieving several goals simultaneously’, in *Machine Intelligence*, eds., E. Elcock and D. Michie, volume 8, (1977).
- [16] H.L.S. Younes, M.L. Littman, D. Weissman, and J. Asmuth, ‘The first probabilistic track of the international planning competition’, *Journal of Artificial Intelligence Research*, **24**, 851–887, (2005).

⁴ <http://www.lond-amb.sthames.nhs.uk/news/performance/performance.html>

Towards a logic of agency and actions with duration

Nicolas Troquard¹Laure Vieu²

Abstract. As far as we know, there is no multi-agent system allowing to talk both about choices of agents or groups of agents, strategies, and about sufficiently rich actions. This paper aims at offering a path towards a new more expressive logical framework by mixing a STIT-like logic of agency with a PDL-like logic of action. We present the syntax and ontological motivations, and we highlight the expressivity of the resulting framework on an example.

1 Introduction

Many domains, e.g., agent interaction or social law modeling, require a good framework for time, agency and action. Time is the basis to express dynamic properties and indeterminacy of the future, agency deals with what agents can bring about and actions are the various ways to bring about some state of affairs. As far as we know, there is no multi-agent system allowing to represent these three domains with sufficient expressivity. In particular, we intend to cover actions that have a duration, and that can be categorized on the basis of properties such as expected effects, temporal or participant structure.

Some existing logics answer to some extent such needs. Concerning pure action, the well-known Propositional Dynamic Logic (**PDL**) is a natural candidate. Nevertheless, it is not suitable neither for group action nor for individual and group agency. The logic of “Seeing To It That” (**STIT**) is a logic of agency embedded in a branching-time framework [1]. This is a logic about choices and strategies for individuals and groups. The core idea of logics of agency is that *acting* is best described by what an agent brings about: at some time, an agent *chooses* to constrain that some proposition is true. However, in some circumstances, not being able to explicitly refer to *actions* remains a weakness. One expresses sentences of the form “Mary sees to it that the coyote is dead” but not “Mary shoots at the coyote” or “Mary poisons the coyote”, i.e., the manner of bringing a state of affairs is out of concern. In addition, in **STIT**, it is generally considered that Mary’s acting does not take time: actions cannot be suspended half-way and one cannot express that an action starts while another is going on. This last point has already been overcome in [3] with the operator *istit*, but this logic still doesn’t involve actions explicitly.

It appears that we need a richer logical system, for reasoning about time, agency and actions with duration. One research avenue is to capitalize on strength of both **PDL** and **STIT**. The aim of this paper is to investigate this avenue, offering an expressive logical framework to support time, agency (for individuals and groups) and actions with duration and other properties, for modeling interactions between agents.

¹ Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier and Laboratorio di Ontologia Applicata - ISTC, Università degli Studi di Trento; troquard@irit.fr

² Institut de Recherche en Informatique de Toulouse, CNRS and Laboratorio di Ontologia Applicata - ISTC, CNR, Trento; vieu@irit.fr

2 Description and justification of the framework

Within the limits of this extended abstract, we give the language of the new logic and describe some elements of its semantics intertwined with ontological justifications. Possible axioms or theorems are proposed in formulas labelled **(n)**. Models have been fully characterized but axiomatization proper is still work in progress.

Language. $\mathcal{A}ct$ is the set of actions, and $\mathcal{A}ct_\lambda := \{\alpha_\lambda \mid \alpha \in \mathcal{A}ct\}$ is the set of continuations of those actions. $\mathcal{A}tm$ is the set of atomic propositions. $\mathcal{A}gt$ is the set of agents. By notational convention, $\alpha \in \mathcal{A}ct$, $\alpha_\lambda \in \mathcal{A}ct_\lambda$, $\beta \in \mathcal{A}ct \cup \mathcal{A}ct_\lambda$, $p \in \mathcal{A}tm$, $a \in \mathcal{A}gt$ and $A \subseteq \mathcal{A}gt$. A formula can have the following syntactic form:

$$\begin{aligned} \varphi &\triangleq \perp \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{S} \varphi \mid \varphi \mathcal{U} \varphi \\ &\quad \mid \Box \varphi \mid [\beta : a] \varphi \mid [\overline{\alpha : a}] \varphi \mid Stit_A \varphi \end{aligned}$$

A model is a tuple $\mathcal{M} = \langle W, <, R_\square, R_{ACT}, agent, v \rangle$, where W is a set of indexes, partially ordered by the strict temporal precedence $<$. Non comparable indexes are grouped into moments by the equivalence relation R_\square . $R_{ACT}(\beta)$ is a function associating an index w to the index where the performance at w of β ends, $agent$ is a function associating to each action its agent, v is the valuation function.

As in **STIT**, \mathcal{S} and \mathcal{U} are the standard since and until temporal operators. Future and past operators are defined as usual: $\mathbf{F}\varphi \equiv \top \mathcal{U} \varphi$ and $\mathbf{P}\varphi \equiv \top \mathcal{S} \varphi$. $\Box \varphi$ stands for historical necessity of φ (φ is true at every index of the moment). \square is an *S5* modal necessity (**1**). Possible readings of $Stit_A \varphi$ are “the group A sees to it that φ ” or “the group A ’s current choice ensures φ whatever other agents do”.

As in **PDL**, $[\beta : a]$ means that a starts performing action β and that φ holds afterwards. $[\overline{\alpha : a}] \varphi$ means that α has just finished and that φ was true before. By tradition, \Diamond will be used as abbreviation for $\neg \Box \neg$ and similarly for $[\beta : a]$ and $[\beta : a]$.

We will illustrate the logic by the example on Figure 1.

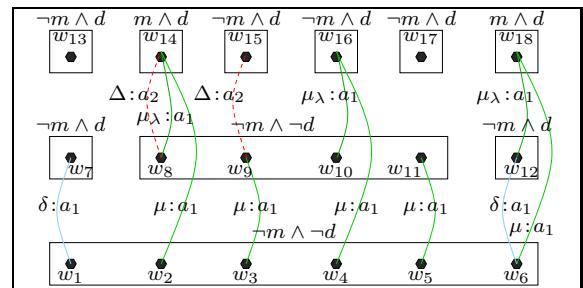


Figure 1. The two cooks. a_1 and a_2 have to prepare a meal. m stands for “the main course is done” and d for “the dessert is done”. δ and Δ are actions of (resp.) a_1 and a_2 cooking the dessert, while μ is the action of a_1 cooking the main course. Boxes are moments containing indexes.

Time. As in **STIT**, our logic assumes a branching time on moments, linear in the past: at each moment, an agent can make different choices, that is, decide to execute different actions bringing about different futures. Maximal linear sets of moments are called

histories; indexes can be seen as moment-history pairs. Models do not constrain all moments to be temporally comparable; an extension could be to do so, adding coincidence of moments through the notion of *instant* as in [1].

Actions. All actions take time ($[\beta : a]\varphi \rightarrow \mathbf{F}\varphi$ (2)). Actions in the present logic are operators thus not properly speaking “events performed by an agent” since events, when they are acknowledged as citizens of the world, are conceived of as concrete individuals, uniquely situated in time and space [2]. These operators correspond to *types*, not tokens, as a given action may occur repeatedly. They are though of a very restricted sort of types. The agent, as well as all other participants, are fixed. The only remaining parameter is time.

Actions correspond to achievements and accomplishments [5], thus two occurrences of the same action cannot overlap ($[\alpha : a]\mathbf{P}[\overline{\alpha : a}]\varphi \rightarrow \mathbf{P}[\overline{\alpha : a}]\varphi$ (3) and $\langle\alpha : a\rangle\top \wedge \mathbf{F}[\alpha : a]\varphi \rightarrow [\alpha : a]\mathbf{F}[\alpha : a]\varphi$ (4) are expected to be valid). Each occurrence runs linearly ($\langle\beta : a\rangle\varphi \leftrightarrow [\beta : a]\varphi$ (5), $\langle\overline{\alpha : a}\rangle\varphi \leftrightarrow [\overline{\alpha : a}]\varphi$ (6), $[\alpha : a][\overline{\alpha : a}]\varphi \rightarrow \varphi$ (7) and $\varphi \wedge \langle\alpha : a\rangle\top \rightarrow [\alpha : a]\overline{[\alpha : a]}\varphi$ (8) are assumed to be valid).

At a same index, more than one action can be performed, by the same or another agent. In the above example, at w_6 , a_1 performs μ and δ ($w_6 \models \langle\mu : a_1\rangle\top \wedge \langle\delta : a_1\rangle\top$).

An action is simply executed or not at an index, but it can unfold into different courses at different indexes of a same moment: in agreement with the STIT approach, actions are not deterministic. In particular, the duration of an action may be left unspecified. That is, not only different occurrences may have different lengths, but the possibly different courses of the same action occurrence may have different lengths on different histories. Action duration can for instance be influenced by the availability of resources. It is also influenced by the fact that actions may be suspended before completion, for reasons external or internal to the agent [3]. Since actions may abort, starting an action does not imply obtaining some expected result: $[\alpha : a]\varphi \rightarrow \square[\alpha : a]\varphi$ is not valid. This means that in our approach, actions are not simply characterized by preconditions and results; we rather focus on the decision of the agent to perform an event of some sort.

Continuation of an action, completed actions. Assuming that actions have a duration and can abort before completion allows to assume that the agent has control over the execution of an action. At each moment during the execution, the agent can decide to keep on performing it or not. On the other hand, in a STIT framework with several agents, agents share the set of indexes, and as a result, whenever an agent makes a choice, all other agents too. This appears too demanding, as simply continuing what has been initiated before is not really a new choice. For both these reasons, we introduce particular actions representing the *continuation of an action*. Introducing in an explicit manner continuations of an action is actually a good way to formalize the notion of *control* on the action [4]. We follow Searle in holding that actions end when the bodily movement is finished, i.e., all actions are totally under control and thus continued up to their end ($\langle\alpha : a\rangle\top \rightarrow \langle\alpha_\lambda : a\rangle\top \wedge \langle\overline{\alpha : a}\rangle\top$ (9) is assumed to be valid). Of course, if a continuation is available, it means that the corresponding action has started before ($[\alpha_\lambda : a]\varphi \rightarrow \mathbf{P}[\alpha : a]\varphi$ (10)). Continuations of a given occurrence of an action do not have a unique starting point, as they are repeated till the end of the action, but they all run till the same ending point (cf. Formula 9).

Since actions can abort, when an action ends, it is not necessarily completed. We thus introduce propositions $\text{comp}(\alpha) \in \text{Atm}$, one for each action α , that reads “action α is completed”. An action α is completed when it has just ended and no continuation is possible; an action aborts if it ends but some continuation is still possible. In

our example, action μ is aborted in w_9 and w_{11} and completed in w_{14}, w_{16} and w_{18} . This notion of completion may be used to express that completed actions do have specific effects; categories of actions could then be introduced on the basis of effects of completed actions. However, one may also want to allow for completed but failed actions, just as in w_{16} , where m doesn’t hold.

Not doing anything. As observed before, STIT’s requiring that if an agent makes a choice at a moment, all other agents too, is too demanding, and needs to be fixed not only for action continuations. In fact, agents may remain simply passive when others really choose to act. To express this, we introduce a set of propositions $\lambda(a) \in \text{Atm}$, one for each agent a , that reads “the agent a remains passive”. An agent a remains passive when it does not perform an action nor a continuation. In the example, agent a_2 remains passive everywhere but w_8 and w_{10} ($w_8 \models \neg\lambda(a_2)$).

Choices and groups agency. We can analyze the combination of choice and action. In multi-agent systems, and particularly in STIT, an agent’s choice is understood as choosing to bring about a state of affairs. In the present work, we handle choice as choosing to perform a set of actions. To deal with agency we still use the *Stit* operator. *Stit* is an *S5* modal operator (11). Moreover, if $A \subseteq B$ then $\text{Stit}_A\varphi \rightarrow \text{Stit}_B\varphi$ (12). We have the interesting property that if a performs α , a sees to it that it performs α , which can be stated by $\langle\alpha : a\rangle\top \leftrightarrow \text{Stit}_a\langle\alpha : a\rangle\top$. Similarly, it is also true that $\lambda(a) \leftrightarrow \text{Stit}_a\lambda(a)$. In the example, at the moment of w_1 , agent a_1 has three choices, corresponding to performing action δ , performing action μ or performing both ($w_2 \models \text{Stit}_{a_1}\langle\mu : a_1\rangle\top \wedge \diamond\text{Stit}_{a_1}\neg\langle\mu : a_1\rangle\top$).

Concerning cooperation, at w_8 none of a_1 and a_2 can see to it that both the main course and the dessert are cooked ($\neg\diamond\text{Stit}_{a_1}(m \wedge d) \wedge \neg\diamond\text{Stit}_{a_2}(m \wedge d)$) but they can cooperate for that ($w_8 \models \text{Stit}_{\{a_1, a_2\}}(m \wedge p)$); it is achieved by means of a_1 continuing to perform μ and a_2 executing Δ .

3 Conclusion

In this work, we have introduced actions with duration, action continuations, and explicit choices of remaining passive in the language of a STIT-like framework. By doing so, we have also cured an annoying feature of STIT which is that when an agent makes a choice, other agents too. Moreover, choices in STIT are arbitrary partitions of moments; we made the notion of choice clearer by constructing choices over sets of actions.

Besides working out a full axiomatization, extensions include: less restricted types of action, expected results, action temporal structure, composing operators and strategies for individuals and groups.

REFERENCES

- [1] N. Belnap, M. Perloff, and M. Xu, *Facing the future: agents and choices in our indeterminist world*, Oxford, 2001.
- [2] *Events*, eds., R. Casati and A. Varzi, Dartmouth Publishing, Aldershot, 1996.
- [3] Thomas Müller, ‘On the formal structure of continuous action’, in *Advances in Modal Logic, Volume 5*, eds., Renate Schmidt, Ian Pratt-Hartmann, Mark Reynolds, and Heinrich Wansing, pp. 191–209. King’s College Publications, (2005).
- [4] J. Searle, *Rationality in Action*, MIT Press, Cambridge, MA, 2001.
- [5] Z. Vendler, ‘Verbs and times’, *Philosophical Review*, **56**, 143–160, (1957).

5. Model Based Reasoning

This page intentionally left blank

Better Debugging through More Abstract Observations

Wolfgang Mayer and Markus Stumptner¹

1 INTRODUCTION

Recent years have seen considerable developments in modelling techniques for automatic fault location in programs. However, much of this research was focussed on exploiting a single property of programs or their execution, introducing increasingly complex observation models to keep diagnosis sets manageable. While this has helped somewhat to improve results, it has also led to an increased burden on the users, which have to answer increasingly complex questions posed by the debugging tool to obtain good results. As a result, an increasing number of questions cannot be answered or are answered incorrectly, leading to poor debugging result.

Instead, we propose to combine assertions about multiple abstract properties of program executions to battle the complexity inherent to specification of program behaviour. Assertions are very simple and cannot contribute much individually, but in combination with an automatic model refinement approach provide a powerful tool to diagnose complex faults in programs.

The use of model-based reasoning approaches for localising faults in various software systems has grown to encompass a variety of systems and languages [4, 11, 8, 1]. Work on model-based debugging of has long limited itself to treating user queries in the same fashion as test vectors. That means the debugger indicated a point during program execution that it was considering (e.g., a particular statement), and then queried whether particular values in the program state at some point during execution were correct or not. In the extreme case it asked the user to provide a correct answer for an incorrect value. In the debugging literature the user's contribution to the debugging process is usually referred to as an "oracle", illustrating the depth of knowledge that the user is expected to possess about the actual workings of the program on a given test case.

Answering these oracle queries posed by the system could be surprisingly difficult, potentially needing the *correct* program behaviour to be hand-simulated up to that point. In [9] it was postulated that for interactive debugging, queries based on high level descriptive predicates, being closer to the definitions used in declarative languages, would be both more powerful and easier for the debugger user to apply.

2 MODEL-BASED DEBUGGING

The key idea of *Model-based Software Debugging (MBSD)* is to compile the program into a set of models which reflect (ab-

stract) properties about the program's behaviour. Further, a set of test cases is assumed to be given, where some or all of the tests exhibit incorrect results. Comparing the differences between the program's behaviour when executed on all test cases and the result and assertions specified by the test cases, possible explanations can be inferred using a variant of Reiter's consistency-based diagnosis framework [10].

Previously, a model based on the well-known interval abstraction [2] to approximate the actual behaviour of the program was proposed. When using only this model, certain faults can be located, but for programs predominantly dealing with nested loops and heap data structures, the number of possible explanations is larger than desirable in practice. Analysis of the results has revealed that many of the unwanted results can be removed by strengthening the reasoning engine to detect inconsistencies more often.

The following section proposes a set of abstract properties that help to strengthen the MBSD framework by asserting aspects of program executions that are orthogonal to the ones derived using the interval abstraction. Through interactions between the different properties, the model's structure and fault assumptions can be refined iteratively.

3 DEBUGGING WITH ABSTRACT PROPERTIES

While there are no intrinsic requirements on the abstractions used to query users, we propose the following collection of properties. We do not claim that these are sufficiently complete to effectively debug any fault or program. Rather, abstractions that are simple to decide for a user without detailed knowledge about the program have been selected such that synergies between abstractions and (abstract) program executions allow for effective debugging.

Formally, all abstractions can be modelled as Predicate Abstraction [5] or lattices [2]. Space limitations preclude us from providing formal definitions and we present textual descriptions only.

Read- and write-only. Observations that all (or a particular subset of) elements of a data structure used as input to a loop or a method call are either only read or updated only. Read-only assertions detect conflicts if the program attempts to modify the structure. Write-only access is used to decouple the previous values of the data structure from the updated values, leading to smaller conflicts in case the update operation is included in a fault candidate.

¹ Advanced Computing Research Centre, University of South Australia. [mayer,mst]@cs.unisa.edu.au

A likely scenario is a user looking at a program execution stopped at a breakpoint in a debugger, looking for a reason why a data structure was updated incorrectly. Noticing that some part of the data structure has changed compared to the last time the program was stopped, but shouldn't have been modified according to the intended operation of the program, the user opens a dialog box, selecting the structure in question, and asserting "should not change between b_1 and b_2 " (assume for now that b_i denote the two breakpoints). The automatic debugger incorporates the assertion into the specification of the program's intended behaviour and eliminates all explanations contradicting the new partial specification. This process is repeated until a the true cause of the update has been isolated.

Region reachability. Statements that should be executed always (or never, at least or at most once) are identified to remove paths that would otherwise contribute to spurious explanations.

While precise specifications of the program's control flow represented as for example linear logic or finite state automata may be difficult to provide for a casual user, developers typically have a vague idea of the intended control flow of a program. Simple observations along the lines of "there shouldn't be an uncaught exception when executing this test case" or "this alternative must not be executed" are easy to determine and to assert. The debugger can utilise this specification to build refined representations of the program's control flow, which may reduce loss of information in other abstractions.

Exhaustive Traversal. Elements of arrays and dynamic data structures are often processed such that either all of them are read or updated. If the underlying data structure is monotonic, this allows to bound the number of iterations a loop is executed, which in turn allows to eliminate spurious diagnoses through improved modelling.

Similar to the previous property, it may be known to the developer that for example each element of a data structure must contribute to a computation, whereas the exact computation is unknown. Specifying that a loop should iterate over the entire structure is easy to do, allowing the debugger to relate the size of the data structure and the number of loop iterations. Subsequently, this information can be used refine the model by e.g. unrolling the loop. Note that this is achieved without writing formal specifications.

Variable (in)dependence. While the precise computations performed by a program may not be known, the flow of and dependencies between values computed is typically much easier to establish. Dependencies between variables can be utilised to infer missing statements or uses of wrong variables [6]. Dependency information could also be used to chose a suitable abstraction for dependent variables. For example using a relational model for local dependent variables may reduce spurious values (and fault candidates) compared to a purely non-relational approach.

Subproblem independence. Loops can be modelled more precisely if it is known that computations in different iterations are independent of each other [7]. Many common algorithms dealing with arrays and lists perform computations that could be carried out independently. If it is known that this is the case

for a particular loop, the loop can be "parallelised", leading to independent control and data flow in distinct iterations, which in turn allows the debugger to compute more precise explanations.

Loop specific invariants. Loops based on counters or other induction variables can often be bounded if monotonicity of the induction variable is assumed. While these properties can often be inferred automatically [3] in case the statements in the loop are assumed correct, in case fault assumptions are present, this may not be possible. Simple program analysis techniques combined with user-specified abstract invariants, such as monotonicity, may help to bound the number of iterations and dependences between variables.

4 Outlook

We have presented an approach to semi-automatic debugging that builds on prior model-based debugging work and extends it by a formal framework to incorporate high-level observations. These use a fixed library of predicates that are expressed in terms of the Abstract Interpretation framework in terms of trace segments. This allows for simpler but powerful user interaction, leading to a more precise approximation of the common grounds between intended and actual program behaviour. While ideas have been shown promising result on a set of toy problems, an implementation that can handle every-day programs is still outstanding. Suitable strategies for displaying and interacting with the users, in particular how to present the findings of our semi-automatic debugger and how to pose queries to the user also remains for further investigation.

REFERENCES

- [1] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. Theseider Dupre, 'Cooperative model-based diagnosis of web services', in *Proc. DX'05 Workshop*, (2005).
- [2] Patrick Cousot and Radhia Cousot, 'Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixpoints', in *POPL'77*, pp. 238–252, (1977).
- [3] Patrick Cousot and Radhia Cousot, 'Abstract interpretation based program testing', in *Proceedings of the SSGRR 2000 Computer & eBusiness International Conference*, (2000).
- [4] Alexander Felfernig and Konstantyn Shchekotykhin, 'Debugging user interface descriptions of knowledge-based recommender applications', in *Proc. 2006 Int'l Conf. on Intelligent User Interfaces*, (2006).
- [5] Susanne Graf and Hassen Saïdi, 'Construction of abstract state graphs with PVS.', in *CAV*, volume 1254 of *LNCS*, pp. 72–83. Springer-Verlag, (1997).
- [6] Daniel Jackson, 'Aspect: Detecting Bugs with Abstract Dependencies', *ACM TOSEM*, 4(2), 109–145, (1995).
- [7] Wolfgang Mayer and Markus Stumptner, 'Approximate modeling for debugging of program loops', in *Proc. DX'04 Workshop*, (2004).
- [8] Wolfgang Mayer and Markus Stumptner, 'Debugging program loops using approximate modeling', in *Proc. ECAI*, (2004).
- [9] Wolfgang Mayer and Markus Stumptner, 'Model-based debugging with high-level observations', in *IFIP International Conference on Intelligent Information Processing (ICIIP)*, (2004).
- [10] Raymond Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, 32(1), 57–95, (1987).
- [11] Franz Wotawa, 'Debugging VHDL designs: Introducing multiple models and first empirical results', *Applied Intelligence*, 21(2), 159–172, (2004).

Logic Profiling for Multicriteria Rating on Web Pages

Alessandra Mileo¹

Abstract. I want to propose a general framework for user-oriented and content-based recommender systems aimed at providing preferential multicriteria rating on Web Pages and automatic user's profile generation and updating through logic programming techniques.

1 Introduction

The World Wide Web (WWW) is constituted by a large, distributed set of heterogeneous documents that is constantly changing, making it harder for a visitor to locate interesting pages exploring all possible paths where pertinent information can be found.

Some of the solutions to this problem proposed so far were mainly based on machine learning techniques [4, 5], sometimes tracing the whole user's browsing behavior to update the profile [3], and generally content-based. Several limitations of these systems showed that multicriteria rating, multidimensionality and flexibility are needed in order to improve the understanding of users and items [1].

2 User's Profile Generation

In my approach, user's browsing behavior, is represented by a set of meaningful user's actions traced in a log file during browsing, and extracted in form of logic predicates when the user asks the system for suggestions on links of a page. User's profile is related to i)sources and ii)reasons to recommend a link.

Interests on sources are represented by a set of pages Url_i (in form of URL addresses) that are interesting to the user. These pages can be a subset of all pages that are included among the *favourites* links in the browser menu and are written as a set of facts of the form $fav(Url_i)$. When a user asks for suggestions, the initial profile is enriched with new facts derived from the *recent* user's browsing behavior. Which actions should be considered meaningful and consequently used to update the profile is a crucial issue, as a user may perform several kinds of actions mostly irrelevant in determining user's preferences in a specific context. Thus, I decided to select a reduced set of actions among those traced in the log file, and rewrite them as logic predicates to be used in the inference process.

The filtering mechanism considers meaningful:

- i. to visit an Url with frequency N for an average dwell time D : $vis(Url, N, D)$;
- ii. to add/remove an Url to/from the list of favourites at time T : $add(Url, Time) / rem(Url, Time)$;
- iii. to follow/discard a suggestion with a frequency N : $foll(Url, N) / disc(Url, N)$.

Any other action is filtered out.

¹ Dipartimento di Informatica e Comunicazione, Università degli studi di Milano. Email: mileo@dico.unimi.it

Preferences on reasons are defined by the user in form of an ordering relation on the reasons for which a page can be recommended.

Each *reason for recommendation* is associated to a function $f_i(P_0, P_j)$ representing a relation among the current page the user is visiting, P_0 , and each of the pages linked by P_0 , namely P_1, \dots, P_j .

In preliminary tests I considered three reasons to recommend a link: f_1 is related to a similarity metric, f_2 expresses how a page is correlated with interesting *sources* and f_3 combines both criteria. According to the ordering relation, each function f_i is associated to a value $p(f_i) \in \{1..i\}$ that will be used for ranking.

3 Updating Profile by Adapting Rules

The formal setting I propose for profile update is that of Answer Set Programming (ASP). ASP is based on the *stable models* semantics for Logic Programs proposed by Gelfond and Lifschitz [2] and it can be seen as bringing together concepts and results from Logic Programming, Default Reasoning and Deductive Databases.

The intuition is that of using automated commonsense (non-monotonic) reasoning to update user's interests by revising the logic program (adding new facts and rules according to information from the log file) and by modifying numerical thresholds in logic rules.

Parameters update allows to adapt the learning process to different users as well as to a user that changes his/her interests.

A page Url_j is classified as being either *interesting* or *uninteresting*. In particular, Url_j is interesting if the log file traced that it has been added to the list of favourites during browsing². Thus, Url_j is interesting either by default (it is in the list of favourites) or if:

- i) Url_j has a frequency N of visits, average dwell time $D > \epsilon$, or
- ii) recommended Url_j has been visited with frequency $N > Min_f$,

When a page is inferred to be uninteresting, the default mechanism detects an *exception to default*. This may happen in three cases:

- i) Url_j has a frequency N of visits, and average dwell time $D < \epsilon$;
- ii) suggested Url_j is ignored with a frequency $N > Min_d$;
- iii) Url_j is removed by favourites at time T , not added again later.

Values for ϵ (expressed in milliseconds), Min_f (real) and Min_d (real) are initially fixed by the system as equal to the minimum value extracted from the log file, and then revised on the basis of how user reacts to suggestions.

4 Computing suggestions

The process of computing suggestions and adding them to a Web Page consists of four phases:

² Supposing a sort of coherence of the search, the log file is cleaned each time the user closes the browser, or manually types a different url address into it, thus moving in a new search context.

1. *Clustering*: page P_0 and pages P_j ($j = 1..m$) pointed by P_0 at the first level of depth, are clusterized on the basis of a similarity function $s(P_0, P_j)$ (e.g. cosine similarity);
2. *Preferred Ranking*: page P_0 and pages P_j are given a weight $W_j = |WC_j| + |WI_j|$ where:

$$WC_j = \{Url_i \mid Url_i \text{ is in the same cluster as } P_j\}$$

$$WI_j = \{Url_i \mid Url_i \text{ points to/is pointed by } P_j\}$$

3. Computing similarities:

- $f_1(P, P_j)$ is equal to the value of the similarity function $s(P_0, P_j)$ if P_0 and P_j are in the same cluster, zero otherwise;
- $f_2(P, P_j)$ is equal to $|WC_j|$ if P_0 and P_j are in the same cluster, zero otherwise;
- $f_3(P, P_j)$ is equal to $W_j - W_0$ if $W_j > W_0$, zero otherwise.

Values of f_i are disposed in a similarity matrix $S \in N^{j \times i}$ where the j -th row corresponds to page P_j , the i -th column corresponds to correlation functions f_i , and each element s_{ji} in the matrix is the result of $f_i(P, P_j)$.

S is multiplied by a *correlation matrix* $C \in R^{i \times i}$ which is a diagonal matrix where each element d_{ii} on the diagonal, corresponds to the weight of f_i according to preferences on reasons expressed by the ordering relation.

By this operation, a *weighted similarity matrix* $S_w \in N^{j \times i}$ is obtained. Each row in S_w is associated a value r_j which is a linear combination of the elements in the j -th row. This value is then used to rank pages.

4. *Adding suggestions*: once r_j has been computed for each page P_j , results are presented to the user by associating symbols to urls in P_0 or by showing an ordered list of the urls of P_0 , accessible by clicking on the correspondent link.

Example 1 Suppose user U is browsing page P_0 , pointing to four pages P_j , $j = 1..4$. Interesting sources are Url_i , $i = 1..4$.

Suppose $s(P_i, P_i) = 1$, $s(P_0, P_3) = 0.92$, $s(P_0, Url_4) = 0.87$, $s(P_1, Url_1) = 0.7$, $s(P_1, Url_2) = 0.8$, $s(P_2, Url_3) = 0.9$, $s(P_3, Url_4) = 0.78$ ³. Values are placed in the correlation matrix S ; weights and links are illustrated in Figure 1.

Suppose we expressed the order $f_1 \succ f_3 \succ f_2$, thus $p(f_1) = 3$, $p(f_3) = 2$ and $p(f_2) = 1$ are placed on the diagonal of matrix C .

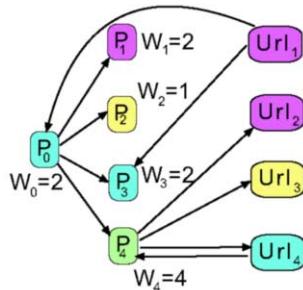


Figure 1. Example 1: preferred ranking

Matrix S_w is given by $S \cdot C$ as follows:

$$S_w = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.92 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2.76 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

The ranking value r_j for page P_j is a linear combination of elements of the j -th row of S_w , i.e. $r_j = \sum_{i=1}^3 s_{ji}$, thus $r_1 = r_2 = 0$, $r_3 = 2.76 + 1 = 3.76$, and $r_4 = 4$.

Note that P_3 is less preferred than P_4 despite its being in the same cluster as P_0 . This is due to the combination of two factors: first, correlation between P_0 and P_3 is not as substantial as the increment of interestingness the user could get in visiting page P_4 ; second, function f_3 combining interestingness of a page and similarity metric, is not much less preferred by the user than f_1 , based on similarity only; as a consequence, due to the preferential nature of the multicriteria framework, page P_3 is slightly less preferred than P_4 because browsing through it results in a significant increment of interestingness for the user, although P_3 is more similar to P_0 than P_4 . A different order on functions f_i would affect this result.

5 Conclusions and Future Work

Although no prototype is available for complete validation yet, preliminary tests on a few log file instances showed that classical machine learning approaches mentioned in the introduction, could give better solutions only when a continuative browsing activity of different users is taken into account. My alternative approach considers the profile as user- and interaction-oriented, in that only recent browsing activity of the user who is running the system is detailed and used for profile update. In this case the cost of using a reasoning system at run-time is not too much to pay for the benefits that may be gained, as we can get acceptable recommendations even after a few user's interactions, by combining rating criteria.

Non intrusiveness is granted by the fact that initial user's interests are extracted almost automatically: no complex statistical information, or features selection are needed.

Correlation functions could potentially be n^4 ; this would generate a n -dimensional matrix expressing the preferential order on functions f_i , thus allowing multicriteria rating by a linear combination of the final results of each criterion.

Nonetheless, more detailed experimental results are needed to evaluate effectiveness of this approach and provide significant empirical data. This aspects will be detailed in a future full paper, where also further extensions of the framework will be investigated, such as i) ordering rules applied to infer interestingness of a page and ii) quantifying interestingness at some degree of a page.

REFERENCES

- [1] G. Adomavicius and A. Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Trans. Knowl. Data Eng.*, **17**, 734–749, (2005).
- [2] M. Gelfond and V. Lifschitz, 'The stable model semantics for logic programming', *ICLP/SLP*, 1070–1080, (1988).
- [3] H. Lieberman, 'Letizia: An agent that assists web browsing', *IJCAI*, **1**, 924–929, (1995).
- [4] D. Mladenić, 'Web location of personal web watcher project', <http://www-ai.ijs.si/DunjaMladenic/pww.html>, (1999).
- [5] J.M. Pazzani and D. Billsus, 'Learning and revising user profiles: The identification of interesting web sites', *Machine Learning*, **3**, 313–331, (1997).

⁴ This number is limited by the space available and may affect performances, but $n > 2$ already allow to apply multicriteria rating.

³ Similarity results for the remaining combination of pages are lower than a fixed minimum, so they are considered equal to zero.

An Empirical Analysis of the Complexity of Model-Based Diagnosis

Gregory Provan¹

Abstract. We empirically study the computational complexity of diagnosing systems with real-world structure. We adopt the structure specified by a small-world network, which is a graphical structure that is common to a wide variety of naturally-occurring systems, ranging from biological systems, the WWW, to human-designed mechanical systems. We randomly generate a suite of digital circuit models with small-world network structure, and show that diagnosing these models is computationally hard.

1 Diagnostic Inference for Complex Systems

The problem of model-based diagnosis (MBD) consists of determining whether an assignment of failure status to a set of mode-variables is consistent with a system description and an observation (e.g., of sensor values). This problem is known to be NP-complete. However, this is a worst-case result, and some NP-complete problems are known to be tractable for particular problem classes. For example, graph colouring, which is NP-complete, has empirically been shown to have run-times that depend on the graph structure [2].

We are interested in the average-case complexity of MBD algorithms on problem instances with real-world structure. At present, it is not known whether MBD is computationally difficult for the “average” real-world system. There has been no systematic study of the complexity of diagnosing real-world problems, and few good benchmarks exist to test this.

We empirically study the complexity of diagnostic inference within a model-based framework using a suite of randomly-generated problem instances, each of which possesses typical real-world properties. Our experimental results show that problems with real-world structural properties are computationally more difficult than problems with regular or random structure, such as would be generated by a typical random-problem generator.

This article makes two main contributions. First, it describes a technique to generate diagnosis models with real-world structural properties. This approach circumvents the difficulty of assembling a large suite of test problems (benchmark models), given that most large diagnosis models tend to be proprietary. It also enables us to control model parameters (and hence analyse specific parameters). Second, we show empirically that diagnosing models with real-world structure is computationally hard. This provides the first clear experimental demonstration of this computational intractability.

2 The Structure of Real-World Problems

Several recent theoretical studies and extensive data analyses have shown that a variety of complex systems, including biological [6],

social [6], and technological [1, 5] systems, share a common underlying structure, which is characterised by a *small world graph*. A small-world graph (SWG) is a complex network in which (a) the nodes form several loosely connected clusters, and (b) every node can be reached from every other by a small number of hops or steps.

We can measure whether a network is a small world or not according to two graph parameters: clustering coefficient and characteristic (mean-shortest) path length [6]. The clustering coefficient, C , is a measure of how clustered, or locally structured, a graph is; this coefficient is an average of how interconnected each agent’s neighbors are. The characteristic path length, L , is the average distance between any two nodes in the network, or more precisely, the average length of the shortest path connecting each pair of nodes.

Several recent studies of technological systems have shown that they all possess small-world topology [1, 5]. In these studies, each technological system is described in graph-theoretic terms, and the underlying topology of the system graph G is studied. For example, for the electronic circuits studied in [5], the vertices of G correspond to electronic components (gates, resistors, capacitors, diodes, etc.), and the edges of G correspond to the connections (or wires) between the components. These circuits comprise both analog and ISCAS’89/ITC’89 benchmark circuits, and all display C and L parameters that are typical of SWG topologies.

3 Model-Based Diagnosis

We can characterise a MBD problem using the triple $\langle COMPS, SD, OBS \rangle$ [7], where:

- $COMPS = \{C_1, \dots, C_m\}$ describes the operating modes of the set of m components into which the system is decomposed.
- SD , or system description, describes the function of the system. This model specifies two types of knowledge, denoted $SD = (\mathcal{S}, \mathcal{B})$, where the system structure, \mathcal{S} , denotes the connections between the components, and the system behaviour, \mathcal{B} , denotes the behaviour of each component.
- OBS , the set of observations, denotes possible sensor measurements, which may be control inputs, outputs or intermediate variable-values.

We adopt a propositional logic framework for our diagnostic models. Component i has associated mode-variable C_i ; C_i can be functioning normally, denoted as $[C_i = OK]$, or can take on a finite set of abnormal behaviours.

MBD inference assumes initially that all components are functioning normally: $[C_i = OK]$, $i = 1, \dots, m$. Diagnosis is necessary when $SD \cup OBS \cup \{[C_i = OK] | C_i \in COMPS\}$ is proved to be inconsistent. Hypothesizing that component i is faulty means switching from $[C_i = OK]$ to $[C_i \neq OK]$. A (minimal) diagnosis is thus a

¹ Computer Science Department, University College Cork, Cork, Ireland, email: g.provan@cs.ucc.ie

(minimal) subset $C' \subseteq COMPS$ such that: $SD \cup OBS \cup \{[C_i = OK] | C_i \in COMPS \setminus C'\} \cup \{[C_i \neq OK] | C_i \in C'\}$ is consistent.

4 Benchmark Model Random Generator

We study a suite of electronic circuits that are constructed from simple gates. The inference complexity of other classes of model with comparable structures will be similar, since it is the system topology (graph width) that is the primary determinant of complexity and the type of constraint defining the system description is only a secondary determinant of complexity [3].

We generate diagnostic models in a three-step process.

1. generate the graph structure G underlying each model;
2. assign components to each node in G for system Ψ ;
3. generate the system description (and fault probabilities) for Ψ .

Generate Graph Structure for G : We generate a small-world-graph (SWG) using the approach of Watts and Strogatz [9]. This approach generates a graph G with a pre-specified degree of randomness that is controlled by a probability $p \in [0, 1]$. $p \approx 0$ corresponds to a regular graph, and $p \approx 1$ corresponds to an Erdos-Renyi random graph; SWGs occur roughly in the range $.01 \leq p \leq .8$, as has been determined by empirically comparing the C and L parameters of generated graphs and actual networks [6]. As noted earlier, a node in G corresponds to a system component, and an edge linking nodes i and j in G corresponds to a wire between components i and j . We randomly assign a set O of nodes to be observable inputs and outputs, choosing $|O|$ based on system size, i.e., $|O| = 0.15n$.

Assign Components to graph G : For our experiments, we use a set of digital comparator components. Given a topology graph G , we associate to each node in G a component, based on the number of incoming arcs for the node. Given a SWG node with i inputs and o outputs, we assign a component, denoted $\Psi_Z(i, o, \mathcal{B}, w)$ where \mathcal{B} defines the behavioural equations of component Z , and w the weights assigned to the failure modes of Z . For example, a j -of- k gate will output t if at least j out of the k inputs are t . Given a node that has q possible components that are suitable, we randomly select a component with probability $\frac{1}{q}$. For example, the single-input nodes correspond to single-input gates (NOT, buffer), and the dual-input nodes correspond to dual-input gates (AND, OR, NAND, NOR, XOR).

Generate the System Description: Given the selected gate, we then generate its normal-mode equations (and potentially failure-mode equations). We randomly select the mode type (of the k possible failure modes) for any gate-model with probability $\frac{1}{k}$. We assign weights to failure-mode values by assuming that normal behaviour is highly-likely, i.e., $Pr\{C_i = OK\} \approx 0.99$, and faulty behaviour is unlikely, i.e., $Pr\{C_i \neq OK\} \approx 0.01$.

5 Experimental Analysis

Given a set of diagnosis models, we studied the inference complexity of each model Ψ by assigning observations (OBS) and then measuring the inference complexity for computing a minimal diagnosis given (Ψ, OBS) .

We have adopted the causal network approach [3] for our experiments. We generated models containing 50, 60, 70 and 80 components for 103 different values of p ranging from 0 to 1.0, to cover the full range of regular, small-world and random graph structures. We used as our measure of inference complexity the sum of all clique-tables in the causal network model, which is a typical complexity measure for this type of model.² Figure 1 shows our results, where

each data point is the average of 300 runs. The plot for each model of size n has the same shape; the shape occurs since regular graphs (p near 0) are computationally simple, small-world graphs are computationally hard (with a complexity peak near $p \approx 0.7$), and random graphs ($p \approx 1$) also are hard, but less hard than the computational peak in the small-world region. The correlation of cluster-size and inference complexity presented here is consonant with the analysis of clique sizes (and corresponding complexity of probabilistic inference), using ISCAS'85 benchmark circuits [4].

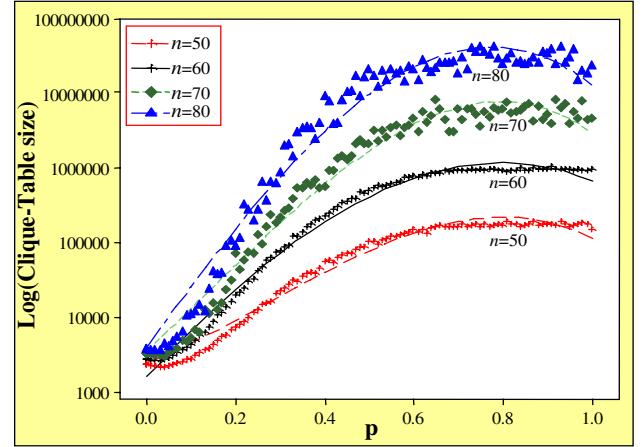


Figure 1. Results of diagnosing SWG-structured digital circuits. Each curve shows a model containing n components.

6 Conclusion

This article has empirically shown that MBD problems with a topology typical of real-world systems, i.e., with a SWG structure, are computationally hard. In fact, over the space of topologies ranging from regular to random, the SWG structure is the computationally least tractable for MBD, a property shared with other inference tasks, such as graph colouring, timetabling and quasi-group analysis [8].

This article has also described a method for generating diagnostic models that have real-world topology and can be tailored to different domains. We have generated models of systems composed of digital circuits, but can generalise this approach to any domain where systems can be composed from a library of components. This method circumvents the problems with using random-graphs for experiments, and provides an alternative to developing suites of hand-built models for benchmarking.

REFERENCES

- [1] Dan Braha and Yaneer Bar-Yam, ‘Topology of large-scale engineering problem-solving networks’, *Physical Review E*, **69**, 016113, (2004).
- [2] Peter Cheeseman, Bob Kanefsky, and William M. Taylor, ‘Where the Really Hard Problems Are’, in *Proc. IJCAI-91*, pp. 331–337, (1991).
- [3] Adnan Darwiche, ‘Model-based diagnosis using structured system descriptions’, *J. Artificial Intelligence Research*, **8**, 165–222, (1998).
- [4] Yousri El Fattah and Rina Dechter, ‘Diagnosing tree-decomposable circuits’, in *IJCAI*, pp. 1742–1749, (1995).
- [5] Ramon Ferrer i Cancho, Christiaan Janssen, and Ricard V. Sole, ‘Topology of technology graphs: Small world patterns in electronic circuits’, *Physical Review E*, **64**(4), 046119, (2001).
- [6] M. E. J. Newman, ‘The structure and function of complex networks’, *SIAM Review*, **45**(2), 167–256, (2003).
- [7] R. Reiter, ‘A Theory of Diagnosis from First Principles’, *Artificial Intelligence*, **32**, 57–96, (1987).
- [8] Toby Walsh, ‘Search in a small world’, in *IJCAI*, pp. 1172–1177, (1999).
- [9] Duncan J. Watts and Steven H. Strogatz, ‘Collective dynamics of “small-world” networks’, *Nature*, **393**, 440–442, (1998).

² This is because the complexity of causal network inference is exponential in the largest clique of the graph (or the graph width) [3].

6. Machine Learning

This page intentionally left blank

Meta-clustering Gene Expression Data with Positive Tensor Factorizations

Liviu Badea¹ and Doina Tilivea¹

1 INTRODUCTION AND MOTIVATION

Although clustering is probably the most frequently used tool for data mining gene expression data, existing clustering approaches face at least one of the following problems in this domain: a huge number of variables (genes) as compared to the number of samples, high noise levels, the inability to naturally deal with overlapping clusters and the difficulty in clustering genes and samples simultaneously. *Fuzzy k-means* or *Nonnegative Matrix Factorization (NMF)* [1] could be used to produce potentially overlapping clusters, but these approaches are affected by a significant problem: the *instability* of the resulting clusters w.r.t. the initialization of the algorithm. This is not surprising if we adopt a unifying view of clustering as a constrained optimization problem, since the fitness landscape of such a complex problem may involve many different local minima into which the algorithm may get caught when started off from different initial states. And although such an instability seems hard to avoid, we may be interested in the clusters that keep reappearing in the majority of the runs of the algorithm. Note that combining clustering results is more complicated than combining classifiers, as it involves solving an additional so-called *cluster correspondence* problem, which amounts to finding the best matches between clusters generated in different runs. The cluster correspondence problem itself could be solved by a suitable *meta-clustering algorithm*.

2 TWO-WAY METACLUSTERING WITH PTF

In this paper we make the simplifying assumption that the overlap of influences (biological processes) is *additive* $X_{sg} \approx \sum_c A_{sc} \cdot S_{cg}$ (1)

where X_{sg} is the expression level of gene g in data sample s , while the expression level of g in s due to biological process c is multiplicatively decomposable into the expression level A_{sc} of the biological process (cluster) c in sample s and the membership degree S_{cg} of gene g in c . Since expression levels and membership degrees cannot be negative: $A_{sc} \geq 0$, $S_{cg} \geq 0$, our clustering problem (1) can be viewed as a *nonnegative factorization* and could be solved using Lee and Seung's seminal *Nonnegative Matrix Factorization (NMF)* algorithm [1]. Such a factorization can be viewed as a "soft" clustering algorithm allowing for *overlapping clusters*, since we may have several significant S_{cg} entries on a given column g of S (a gene g may "belong" to several clusters c).

Allowing for cluster overlap alleviates but does not completely eliminate the instability of clustering, since the NMF algorithm produces different factorizations (biclusters) $(A^{(i)}, S^{(i)})$ for different

initializations, so meta-clustering the resulting "soft" clusters might be needed to obtain a more stable set of clusters.

In this paper, we show that a generalization of NMF called Positive Tensor Factorization (PTF) [2] is precisely the tool needed for meta-clustering "soft", potentially overlapping *biclusters* obtained by NMF object-level clustering. This unified approach solves in an elegant manner both the clustering and the cluster correspondence problem. More precisely, we first run NMF as object-level clustering r times: $X \approx A^{(i)} \cdot S^{(i)}$. (To allow the comparison of membership degrees S_{cg} for different clusters c , we scale the rows of $S^{(i)}$ to unit norm by taking advantage of the scaling invariance of the factorization.) Next, we *meta-cluster* the resulting *biclusters* $(A^{(i)}, S^{(i)})$. This is in contrast with as far as we know all existing (*one-way*) meta-clustering approaches, which take only one dimension into account and fail whenever two clusters correspond to very similar sets of genes, while differing along the sample dimension. The following *Positive Tensor Factorization (PTF)* of the biclusters $(A^{(i)}, S^{(i)})$ represents a *two-way* meta-clustering:

$$A_{s(ic)} \cdot S_{(ic)g} \approx \sum_{k=1}^{n_e} \alpha_{(ic)k} \cdot \beta_{sk} \cdot \gamma_{kg} \quad (2)$$

where k are metacluster indices. (To simplify the notation, we merged the indices i and c into a single index (ic) .)

The columns β_k of β and the corresponding rows γ_k of γ make up a *base set of bicluster prototypes* $\beta_k \gamma_k$, out of which all biclusters of all individual runs can be recomposed, while α encodes the (*bi*)*cluster-metacluster correspondence*. Instead of a perfect bicluster correspondence, we settle for a weaker one (2) in which the rows of α can contain several significant entries, so that all biclusters $A_c^{(i)} \cdot S_c^{(i)}$ are recovered as *combinations* of bicluster prototypes $\beta_k \gamma_k$. The nonnegativity constraints of PTF meta-clustering are essential for obtaining sparse factorizations. (Experimentally, the rows of α tend to contain typically one or only very few significant entries.) The factorization (2) can be computed using the following multiplicative update rules:

$$\begin{aligned} \alpha &\leftarrow \alpha * \frac{(A^T \cdot \beta) * (S \cdot \gamma^T)}{\alpha \cdot [(\beta^T \cdot \beta) * (\gamma \cdot \gamma^T)]} \\ \beta &\leftarrow \beta * \frac{A \cdot [\alpha * (S \cdot \gamma^T)]}{\beta \cdot [(\alpha^T \cdot \alpha) * (\gamma \cdot \gamma^T)]} \quad \gamma \leftarrow \gamma * \frac{[\alpha * (A^T \cdot \beta)]^T \cdot S}{[(\alpha^T \cdot \alpha) * (\beta^T \cdot \beta)]^T \cdot \gamma} \end{aligned}$$

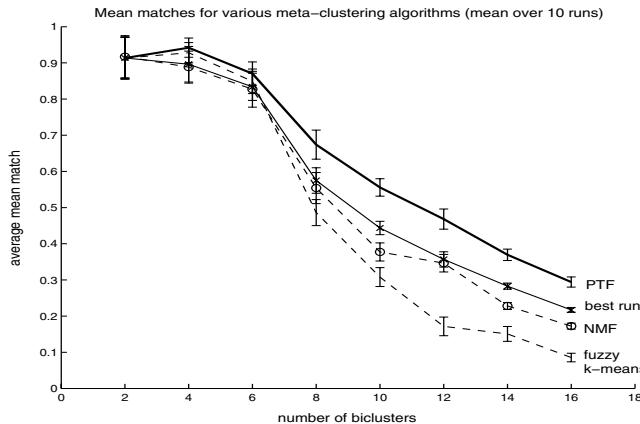
where '*' and '—' denote element-wise multiplication and division of matrices, while '.' is ordinary matrix multiplication.

After convergence of the PTF update rules, we make the prototype gene clusters directly comparable to each other by normalizing the rows of γ to unit norm, as well as the columns of α such that $\sum_{i,c} \alpha_{(ic)k} = r$ and then run NMF initialized with (β, γ) to produce the final factorization $X \approx A \cdot S$.

¹ AI group, National Institute for Research and Development in Informatics, 8-10 Averescu Blvd. Bucharest, Romania. E-mail: badea@ici.ro.

3 EVALUATION ON SYNTHETIC DATA

We evaluated our algorithm on synthetic datasets that match as closely as possible real microarray data. Clusters were modelled using a hidden-variable model $X = A \cdot S + \epsilon$, in which each hidden variable A_c corresponds to the cluster of genes influenced by A_c . We sampled the hidden variables from a log₂-normal distribution with parameters $\mu=2$, $\sigma=0.5$, while the influence coefficients S_{cg} between hidden and observable variables were sampled from a uniform distribution over the interval [1,2]. Finally, we added log₂-normally distributed noise ϵ with parameters $\mu_{noise}=0$, $\sigma_{noise}=0.5$. We chose problem dimensions of the order of our real-world application: $n_{samples}=50$, $n_{genes}=100$, number of genes (respectively samples) per cluster 30 (respectively 15). We compared 4 meta-clustering algorithms (fuzzy k-means, NMF, PTF and the best run) over 10 object-level NMF clustering runs. (Other object level clustering methods perform very poorly and are not shown here). Although all algorithms produce quite low relative errors (under 16%, except for fuzzy k-means which misbehaves for large numbers of clusters), they behave quite differently when it comes to recovering the original clusters. In a certain way, the match of the recovered clusters with the original ones is more important than the relative error. Defining the *match* between two sets of possibly overlapping clusters is nontrivial. For each cluster C_1 from clustering 1, we determine the single cluster C_2 from clustering 2 into which it is best included, i.e. the one with the largest $|C_1 \cap C_2|/|C_1|$. We proceed analogously for the clusters C_2 from clustering 2. Then, for each cluster C_1 (from clustering 1), we determine its match $|C_1 \cap C_2|/|C_1 \cup C_2|$ with the union C_2 of clusters from clustering 2, for which C_1 is the best including cluster (as determined in the previous step). Similarly, we determine matches for clusters C_2 from clustering 2. The average match of the two clusterings is then the mean of all these matches (for all C_1 and all C_2).

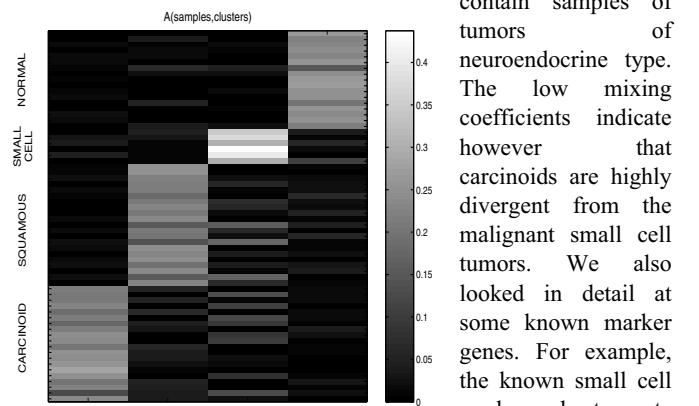


The Figure above shows that PTF consistently outperforms the other meta-clustering algorithms in terms of recovering the original clusters. Note that since clusters were generated randomly, their overlap increases with their number, so it is increasingly difficult for the meta-clustering algorithm to discern between them, leading to a decreasing match. We also observed an inverse correlation between bicluster overlap and matches (Pearson correlation coefficient -0.92). Among all *object-level* clustering algorithms tried (k-means, fuzzy k-means and NMF), only NMF behaves consistently well.

4 METACLUSTERING A LUNG CANCER GENE EXPRESSION DATASET

In the following we show that metaclustering is successful at biclustering a large lung cancer dataset from the Meyerson lab [3], containing 186 lung tumor samples (139 adenocarcinomas, 21 squamous cell lung carcinomas, 6 small cell lung cancers, 20 pulmonary carcinoids) and 17 normal lung samples. For testing our metaclustering algorithm, we first selected a subset of genes (251) that are differentially expressed between the classes (using a SNR measure). More precisely, we selected the genes with an average expression level over 100 and $|SNR| > 2$ for at least one of the classes. Since adenocarcinoma subclasses are poorly understood at the molecular level, we discarded the adeno samples from the dataset and used the histological classification of samples provided in the supplementary material to the original paper [3] as a gold standard for the evaluation of the biclustering results. To eliminate the bias towards genes with high expression values, all genes were scaled to equal norms. Since nonnegative factorizations like NMF cannot directly account for gene down-regulation, we extended the gene expression matrix with new “down-regulated genes” $g' = pos(mean(g_{normal}) - g)$ associated to the original genes g , where $mean(g_{normal})$ is the average of the gene over the *normal* samples and $pos(\cdot)$ is the step function. We then used our PTF metaclustering algorithm to factorize the extended gene expression matrix into 4 clusters with 20 NMF runs.

The algorithm recovered the sample clusters with high accuracy, as can be seen in the following Figure. Note that the overlap between the small cell and carcinoid sample clusters (columns 3 and 1 of A in the Figure) has a biological interpretation: both



1 is *specific* to the small cell cluster, while keratin 5 is specific to the squamous cluster. On the other hand, known proliferative markers like *PCNA* (proliferating cell nuclear antigen), *MCM2* and *MCM6* are *common* to the small cell and squamous clusters, as expected. Overall, our metaclustering algorithm proved quite robust at rediscovering the known histological classification of the various lung cancer types in the Meyerson dataset.

REFERENCES

- Lee D.D., H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, vol. 401, no. 6755, pp. 788-791, 1999.
- Welling M., Weber M. Positive tensor factorization. *Pattern Recognition Letters* 22(12): 1255-1261 (2001).
- Bhattacharjee et al. Classification of human lung carcinomas by mRNA expression profiling... *PNAS* Nov. 20; 98(24):13790-5, 2001.

Identifying Inter-Domain Similarities through Content-Based Analysis of Hierarchical Web-Directories

Shlomo Berkovsky¹, Dan Goldwasser¹, Tsvi Kuflik¹ and Francesco Ricci²

Abstract. Providing accurate personalized information services to the users requires knowing their interests and needs, as defined by their User Models (UMs). Since the quality of the personalization depends on the richness of the UMs, services would benefit from enriching their UMs through importing and aggregating partial UMs built by other services from relatively similar domains. The obvious question is how to determine the similarity of domains? This paper proposes to compute inter-domain similarities by exploiting well-known Information Retrieval techniques for comparing textual contents of the Web-sites, classified under the domain nodes in Web-directories. Initial experiments validate feasibility of the proposed approach and raise open research questions.

1 INTRODUCTION

Providing accurate personalized services to the users requires modeling their preferences, interests and needs. This data is referred in the literature as the User Model (UM) [3]. Typically, service providers build and maintain proprietary UMs, tailored to the domain of the service and UMs representation, dictated by the personalization technique being exploited. Since the quality of the provided personalization heavily depends on a richness of the UMs, different services would benefit from enriching their UMs through importing and aggregating partial UMs, i.e., the UMs built by other, possibly related, services. This will bootstrap the UMs of the services where no UMs exist and enrich already existing UMs, leveraging the quality of the provided personalization [1].

One of the major issues that should be resolved to facilitate proper aggregation of partial UMs is "*Which services can provide valuable partial UMs?*". Considering a wide variety of application domains, we conjecture that in addition to the services from the same application domain, also the services from other (relatively similar) domains can provide valuable partial UMs. However, this inherently brings a question of "*Which application domains are considered as similar domains?*".

This work proposes an automatic approach for devising inter-domain similarities through content-based analysis of the Web-sites, classified under the application domains. Such classifications can be found in various human-edited Web-directories, e.g., Google Directory (<http://directory.google.com>), Yahoo! Directory (<http://dir.yahoo.com>), or Open Directory (<http://dmoz.org>). In Web-directories, the Web-sites are classified under application domains, which are recursively partitioned to more specific sub-domains, and so forth. Since Web-directories are edited manually by human experts, we assume that classification of the Web-sites under domains and sub-domains is correct and that large enough set of Web-sites, classified under a certain domain, can be considered as a reliable representative of the domain contents.

Hence, we base inter-domain similarity computation on a comparison of the textual contents of the Web-sites contained in each domain. For this, we first model the contents of the classified domain Web-sites using well-known Information Retrieval (IR) in-

dexing technique of TF-IDF [4]. It allows representing a domain as a ranked vector of terms (derived from the classified Web-sites), where domain-specific terms are assigned higher weights. Then, inter-domain similarity is computed using a cosine similarity metric [4], where the similarity of two vectors is computed as the cosine of the angle between them in a multi-dimensional space. Initial experimental results, conducted over two Web-directories validate the feasibility of the proposed inter-domain similarity computation approach and raise open questions for future research.

The issue of computing similarity over a hierarchical domains structure is elaborately discussed in [2]. That paper presents, analyzes and experimentally compares a set of similarity metrics over a hierarchical tree of domains, while focusing on exploiting the hierarchical structure as an indicator for the similarity values. Conversely, in this work we aim at computing domains similarity through content-based analysis of the Web-sites, classified to the nodes of Web-directories, whereas hierarchical structure of the directories can serve as a heuristic limitation for the search process.

2 ANALYZING DIRECTORIES' CONTENTS

We suggest calculating inter-domain similarity basing on the textual contents of the Web-sites classified under the domain node in a Web-directory. The proposed approach is based on an inherent assumption that the textual contents reliably represent the domain. Although this assumption is not always true (in addition to the textual contents, nowadays Web-sites contain various graphical, audio and video objects, and use dynamic Web-technologies to generate their contents), we believe that large enough number of classified Web-sites will provide an accurate domain representation and a stable basis for the similarity computation.

To represent textual contents of the Web-sites classified under domains of a Web-directory, we propose to exploit well-known IR indexing techniques. For the indexing, stop-words (such as *and*, *to*, *the*, etc...) and HTML tags are filtered-out, the terms are lemmatized to their basic grammatical forms, and weighted according to their domain-related TF-IDF values, assigning higher weights to domain-specific terms (i.e., terms that are frequent in the domain Web-sites only) [4]. As a result, a ranked vector of weighted domain terms, representing the contents of the Web-sites under a given application domain, is obtained. For example, consider a vector of the top terms in *football news* domain, shown in Table 1. It can be clearly seen that the list reliably represents football terms.

Table 1. Top-ranked terms in football news domain

term	score	high	senator	gameface	rumor	league	archive
weight	462.56	397.14	353.94	320.38	251.88	128.31	103.16

After the domain representative vectors are constructed, inter-domain similarity can be easily computed using one of the existing similarity metrics. In this work, we used cosine similarity metrics, defining the similarity between two vectors as the cosine of the angle between them in a multi-dimensional space:

¹ University of Haifa, Haifa, Israel

² ITC-irst, Trento, Italy

$$\text{sim}(V_x, V_y) = \frac{V_x \cdot V_y}{\|V_x\|_2 \times \|V_y\|_2}$$

where \cdot denotes the dot product between the vectors, and $\|V_i\|_2$ denotes the 2-norm of the vector (i.e., the square root of the sum of the squares of the vector elements). The result of the cosine similarity computation is a single scalar, reflecting the similarity of the respective domains, based on their textual contents.

To obtain the required inter-domain similarity values, the above process of similarity computations should be repeated for all the possible pairs of domains. However, compound structure of nowadays Web-directories and high number of the existing domains, make this task expensive and pose a need for heuristically limiting the search space. We propose to exploit the inherent hierarchical structure of the Web-directories for this purpose. Since the Web-directories are organized as linked hierarchical structures, the search (and similarity computations) may be heuristically limited to a set of relatively close nodes only. For example, a search for the domains, similar to *Arts*→*Movies*→*Genres*→*Drama*, may be limited to other movies genres *Arts*→*Movies*→*Genres*→*, other sub-domains of movies *Arts*→*Movies*→*, and more abstract higher-level domain of visual arts *Art*→*Visual Arts*→*.

3 EVALUATION AND FUTURE RESEARCH

We have implemented a prototype system for inter-domain similarity computations. To obtain the contents of the domain Web-sites, we exploited a simple Web-crawler, configured to download the sites' textual contents only. The contents of the domains were represented using IR indexing techniques employed by Lucene, open-source search engine (<http://lucene.apache.org>). The experiments were conducted over Google Directory and Open Directory.

The first experiment was designed to validate the proposed approach for devising inter-domain similarity values. For this, we downloaded the contents of a small set of application domains and computed their pair-wise inter-domain similarities using the above two Web-directories. The pairs of domains whose similarity was computed were: (a) *Arts*→*Television*→*News*→*News on the Internet* vs. *Arts*→*Television*→*News*→*Sports* (b) *Arts*→*Television*→*News* vs. *Sports*→*Football*→*NCAA*, and (c) *Computers*→*Hardware*→*Storage* vs. *Sports*→*Football*→*NCAA*. Note that these domains were selected as their similarity classification can be easily done by a common sense. Pair (a) is a relatively similar one, since the nodes are siblings and many news Web-sites have a dedicated sports section. Pair (b) is similar to some extent only. Despite the fact that the nodes are distinct in Web-directories, news domain does contain some NCAA news items, however, it contains also many other news topics. Pair (c) is highly dissimilar, as the terminology used in hardware storage devices' domain and in NCAA football is very different. Table 2 summarizes the experimental results.

Table 2. Inter-domain similarities in different Web-directories

	(a)	(b)	(c)
Google Dir.	0.3826	0.2676	0.1416
Open Dir.	0.4205	0.1879	0.1103

Experimental results validate feasibility of the proposed approach, as the similarity values for pair (a) are higher than for pair (b), which, in turn, is higher than for pair (c). However, the results raise a question regarding the stability of the proposed approach with the number of the Web-sites that are indexed. Both Google Directory and Open Directory are structured as highly connected graphs, where each domain comprises a set of sub-domains, whereas the sub-domains of different domains may be interlinked. Since a Web-site may be classified to one of the sub-domains of a

given domain, the second experiment was designed to check the impact of indexing and computing TF-IDF representation of a larger set of Web-sites, rather than the sites classified directly under the node of a given domain. We compared three expansion policies: (i) considering also the Web-sites classified under the sub-domain nodes of a given domain, (ii) considering also the Web-sites pointed by the outgoing links from the Web-sites classified under the node of a given domain, i.e., increasing the depth of the search on the Web, and not in the Web-directory, and (iii) integrating (i) and (ii). These policies were employed on the above three pairs of domains and the similarity computations were repeated for a larger set of Web-sites. Table 3 summarizes the results.

Table 3. Inter-domain similarities using different expansion policies

	(a)	(b)	(c)
Google Dir.	(i) 0.3555	0.4279	0.2709
	(ii) 0.3632	0.2275	0.1829
	(iii) 0.4189	0.3874	0.2888
Open Dir.	(i) 0.2703	0.4929	0.1045
	(ii) 0.3354	0.2164	0.0841
	(iii) 0.3279	0.3372	0.1232

The impact of policy (i) can be defined as negative, as the similarity of the of similar domains in pair (a) decreases, while the similarity of less similar pair (b) and (c) increases. We hypothesize that this can be explained by the fact that indexing the Web-sites of the sub-domain nodes inserts a decent noise to the TF-IDF vectors, as also less similar Web-sites are indexed and graph structure of Web-directories brings in sub-domains, which are actually sub-domains of other dissimilar domains. The same observation is true also for policy (ii) as the similarity of (a) decreases, while the similarity of (b) and (c) increases. However, the impact of (ii) is significantly weaker, and (a) is still more similar than (b), which is more similar than (c). This allows us to conclude that indexing the Web-sites pointed by the outgoing links inserts less noise than indexing the Web-sites classified under the sub-domains of a given domain. The impact of (iii) is unclear and it depends on the specific application domain and additional Web-sites that are indexed.

Although the experiments initially validate the feasibility of the proposed inter-domain similarity computation approach, elaborate experiments and analysis are still required. In the future, we plan to analyze statistical properties of the domains for concluding regarding the conditions and domains, where the proposed approach is applicable. We plan to study the use of the graph structure of Web-directories for heuristically limiting the similar domains search. For this, we will check the correlation between the vicinity of nodes in the Web-directories' graph and their content-based similarity. Also, we plan to check the feasibility of the above heuristics in sparse application domains with a small number of classified Web-sites.

As these questions are answered, we plan to investigate the task of aggregating the partial UMs. We believe that the proposed similarity computation approach can be a stable basis for a dynamic user modeling mechanism, facilitating efficient provision of accurate personalization services.

REFERENCES

- [1] S.Berkovsky, "Ubiquitous User Modeling in Recommender Systems", in proceedings of the UM Conference, Edinburgh, 2005.
- [2] P.Ganesan, H.Garcia-Molina, J.Widom, "Exploiting Hierarchical Domain Structure to Compute Similarity", in ACM Transactions on Information Systems, vol.21(1), pp.64-93, 2003.
- [3] A.Kobsa, "Generic User Modeling Systems", in User Modeling and User-Adapted Interaction, vol. 11(1-2), pp.49-63, 2001.
- [4] G.Salton, M.McGill, "Introduction to Modern Information Retrieval", McGraw-Hill Publishing, 1983.

Calibrating Probability Density Forecasts with Multi-objective Search

Michael Carney and Pádraig Cunningham¹

Abstract. The two objectives when estimating the conditional density function in a regression problem are to maximise *sharpness* (the density rewarded to the actual observation), while maintaining *calibration* (the empirical validity of the probability estimates). In this paper we outline a process of optimisation that maximises both these objectives simultaneously to make better probabilistic predictions.

1 INTRODUCTION

Regression is a supervised learning problem where the fundamental task is to predict some *continuous* variable. Formally, the regression problem is the process of estimating the parameters for a model given a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^m$, where the target variable is $t_i \in \mathbb{R}$. Typically, to solve this problem point forecasting methods are used to estimate, $\langle t_i | \mathbf{x}_i \rangle$, the conditional mean of the target given an input pattern. Density forecasting is an important subfield of regression that tackles the practical problem of uncertainty in predictions by attempting to estimate, $p(t_i | \mathbf{x}_i)$, the conditional probability density of the target. This is useful because prediction users are generally sensitive to the possible variance around a prediction. Furthermore, with a correctly specified density forecast they can make the *optimal* decision under uncertainty.

The most common approach to parameter estimation of regression models is maximum likelihood. Density forecasting is not an exception, using a generalisation of maximum likelihood called negative log-likelihood (*NLL*). The convenience of this approach is that the traditional optimisation techniques such as *conjugate gradient* can be used with minimal adaptations. However, on closer examination of this approach, it can be shown that certain aspects of the problem of density forecasting are not addressed and so this technique can lead to poor, sub-optimal, and often misleading models.

In this paper we introduce a new multi-objective approach to optimisation of density forecasting models. We outline a broad framework, based on a multi-objective evolutionary algorithm (MOEA), that can be used to optimise most density forecasting models.

2 GOALS OF DENSITY FORECASTING

The *NLL* addresses the goal of sharpness by rewarding models based on the density of the prediction at the target. The formula for *NLL* is:

$$NLL_i = -\log(p(t_i | \mathbf{x}_i)) \quad (1)$$

Optimising a model using *NLL* is an attempt to find the set of model parameters that maximise the probability density at the target. This approach produces accurate estimates of the conditional

density function in situations where the underlying data generating process (DGP) is known or can be approximated well by the distribution assumed by the model. In circumstances where the DGP is very complex, or the incorrect assumptions are made about the underlying distribution, or over-fitting is an issue, the *NLL* performs poorly, particularly in terms of calibration. Demonstrations of these effects are shown in [3].

Calibration refers to the property that if a predicted density function suggests P percent probability of occurrence, the event truly ought to have probability P of occurring. [6] shows that in classification calibration can be achieved simply; however, in regression the problem is somewhat more difficult and a straightforward post training transformation does not work. In regression, evaluation of calibration is dependent on the assumption that you are attempting to find the model that correctly describes the DGP. In the case where the correct DGP is described, the set of cumulative densities at the targets will be uniform. Therefore, to determine calibration you must carry out the following transformation;

$$z_i = \int_{-\infty}^{t_i} p(u | \mathbf{x}_i) du \quad (2)$$

where z_i is the cumulative of the predicted density at the target t_i . This is known as the Probability Integral Transform (PIT). Diebold et al. [4] show that the z series should be $\{z_i\}_{i=1}^m \stackrel{iid}{\sim} U[0, 1]$.

Typically, it is a histogram of the z values that is displayed (see Fig 2), and the calibration of the model is assessed visually. However, we suggest that a ranking of density models can be obtained by determining the goodness-of-fit of the z values to the uniform distribution. We compared a number of goodness-of-fit tests and concluded that the Anderson-Darling (A^2) [1] test for uniformity was the most robust among the most common tests for uniformity [3]. The A^2 test is negatively oriented returning a 0 in the case where a model is perfectly calibrated. The formula for A^2 is;

$$A^2 = -m - \frac{1}{m} \sum_{j=1}^m (2j-1)[\log(z_j) + \log(1-z_{m-j})] \quad (3)$$

and the z values are sorted in ascending order.

3 OPTIMISATION FRAMEWORK

In the preceding sections we have mentioned two quality scoring metrics for density forecasting, A^2 and *NLL* (calibration and sharpness). Therefore, implicitly we have described a multi-objective optimisation problem. The preferable approach to solve a multi-objective search problem is to use an *a posteriori* MOEA. In the context of

¹ Trinity College Dublin, Ireland, email: firstname.surname@cs.tcd.ie

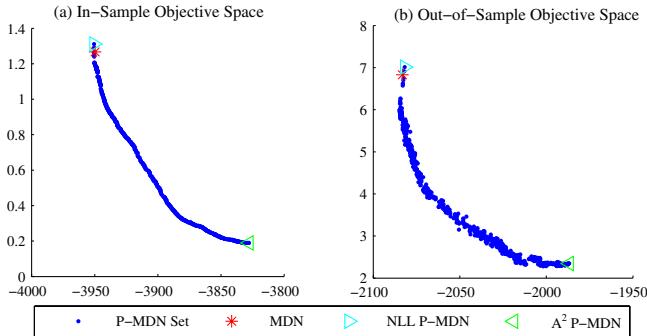


Figure 1. In and out-of-sample objective spaces of Pareto-MDN

MOEA's, *a posteriori* means that the optimisation process maintains an archive of optimal trade-off (non-dominated) solutions, known as the Pareto front, throughout training and the user selects the model that best optimises their goals from the resulting Pareto front of solutions.

Before optimisation the modeler must: (1) Determine a vector representation for the parameters of the density forecasting model. (2) Be able to calculate the A^2 and NLL score for the model's predicted densities. (3) Decide on a mutation and selection strategy for the evolutionary algorithm. Then optimisation can proceed as normal. For further information on the MOEA algorithm, see [3].

Pareto Mixture Density Network A Mixture Density Network (MDN) [2] is a particular class of neural network adapted to produce density forecasts. MDNs represent the conditional density function by a weighted mixture of Gaussians known as a Gaussian Mixture Model (GMM). MDNs are generally optimised using a conjugate gradient technique, however, like most neural networks they can also be optimised using an evolutionary algorithm. Recently, Fieldsend et al. introduced a technique for multi-objective optimisation of neural networks [5]. We have adapted this to fit our framework by using an MDN as the underlying model and setting the objective functions to the NLL and A^2 scores. We use an Evolutionary Strategy (ES) for optimisation of the MDN [5]. This results in objective function and network architecture optimisation. For a full description of the Pareto-MDN see [3].

4 CASE STUDY: FINANCIAL DATA

In this case study, we analyse the performance of an MDN model on estimating the conditional density of foreign exchange data with the purpose of determining a measure of Value-at-Risk. The data is comprised of 1,501 examples of daily price observations for the DeutscheMark/British Pound foreign exchange rate, from April 1985 to January 1992. We transform the daily prices into a log returns series by $r_i = \log(\frac{p_{i+1}}{p_i})$, where p_i is the price at interval i . The return series was separated into a training set of the first 1,000 observations and a test set comprised of the last 500 observations.

The MDN was given 6 hidden units and outputs were represented by a 2 component GMM. An initial model was trained on the data using a standard optimisation technique, in this case we use a Scaled Conjugate Gradient (SCG) method. We use this model as our initial starting position in parameter space for our evolutionary algorithm. We carried out 10,000 iterations of the ES algorithm resulting in a set of 736 non-dominated individuals. Figure 1 shows the in-sample and

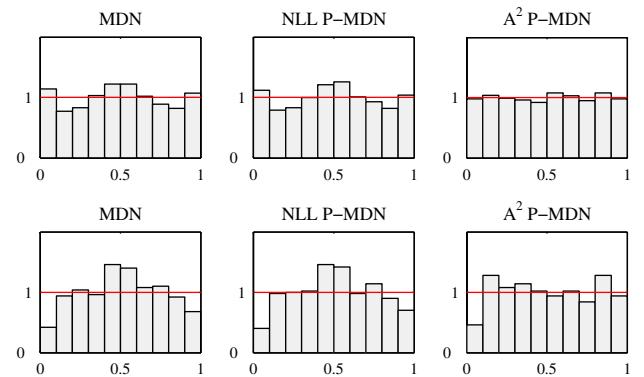


Figure 2. PIT histograms for the training (top row) and test data (bottom).

out-of-sample objective spaces for all individuals in the final Pareto set. Each point on the objective space represents the error scores obtained by one of the Pareto set models. We have highlighted some models of interest, the *MDN* model our benchmark model trained using SCG. *NLL P-MDN* is the model from the Pareto set that has the best in-sample *NLL* score and *A² P-MDN* is the model with best *A²* score.

The Spearman rank correlation coefficient between the in-sample and out-of-sample *NLL* is 0.8589 and *A²* is 0.9848. This suggests little over-fitting of the models to the data on either objective function.

Figure 2 displays the PIT histograms of the z values for the models identified in the objective space. The model with best calibration is *A² P-MDN* in and out-of-sample. The *MDN* and *NLL P-MDN* models show over-confidence in-sample and under-confidence out-of-sample. The effect of this improvement means that for financial applications such as risk management, where rare events are significant, better and more accurate estimates of risk can be reported.

5 CONCLUSIONS

We have introduced and evaluated a multi-objective approach to density forecasting that improves calibration. This is a general method that can be applied to any density forecasting model that can be appropriately parameterised. For example, in [3] we apply this approach to a GARCH model.

REFERENCES

- [1] TW Anderson and DA Darling, 'A test of goodness of fit', *Journal of the American Stat Association*, **49**, 765–769, (1954).
- [2] C.M. Bishop, 'Mixture density networks', Technical report, Neural Computing Research Group, Aston University, Birmingham, (1994).
- [3] M Carney and P Cunningham, 'Calibrating probability density forecasts with multi-objective search', Technical Report TCD-CS-2006-07, Comp Sci Dept, TCD, (2006).
- [4] FX Diebold, T Gunther, and A Tay, 'Evaluating density forecasts with applications to financial risk management', *Inter Econ Review*, **39**(4), 863–83, (1998).
- [5] J Fieldsend and S Singh, 'Pareto evolutionary neural networks.', *IEEE Trans. Neural Networks*, **16**(2), 338–354, (2005).
- [6] A Niculescu-Mizil and R Caruana, 'Predicting good probabilities with supervised learning', in *ICML '05*, (2005).

Term-Weighting in Information Retrieval using Genetic Programming: A three stage process

Ronan Cummins and Colm O'Riordan ¹

1 Introduction

This paper presents term-weighting schemes that have been evolved using genetic programming in an adhoc Information Retrieval model. We create an entire term-weighting scheme by firstly assuming that term-weighting schemes contain a global part, a term-frequency influence part and a normalisation part. By separating the problem into three distinct phases we reduce the search space and ease the analysis of the schemes generated by the process.

Evolutionary computation techniques are proving to be a viable alternative to other standard analytical methods in many areas of IR. Genetic Programming (GP) [2] is an automated searching algorithm inspired by biological evolution. GP has been shown to be an effective approach to learning term-weighting schemes in IR [5]. Firstly, we evolve weighting schemes in a global domain which promote the best terms to use in distinguishing documents. Then, using a suitable global scheme, we evolve term-frequency influence schemes which uses the within-document term-frequency to correctly weight the term-frequency factor. Finally, we evolve normalisation schemes based on the best performing combined global and term-frequency scheme. This framework is an extension of work carried out in [1]. Most term-weighting schemes combine these three aspects to weight query terms and thus score a document in relation to a query.

2 Experimental Framework

The global (gw_t) and normalised term-frequency (ntf) weighting schemes are evolved in a term-weighting function which scores a document (d) in relation to a query (q) as follows:

$$score(d, q) = \sum_{t \in q \cap d} (ntf \times gw_t \times qtf) \quad (1)$$

where qtf is the actual term-frequency of term t in the query. It can be seen that both BM25 [3] and the pivoted normalisation scheme [4] fit this type of model. Tables 1, 2 and 3 show the terminals sets and some GP parameter details for the experiments. The set of functions used for all experiments is $F = \{\times, +, -, /, log, square, square-root\}$. We use an elitist GP strategy and 4% mutation for all experiments. Mean average precision (MAP) is used as the fitness function in all experiments. All solutions are limited to a depth of 6.

2.1 Document Test Collections

The training set for the global problem consisted of 35,000 OHSUMED documents from 1998 and the 63 topics. The training set

Table 1. Global Weighting Problem Terminals

Terminal	Description
N	no. of documents in the collection
df	document frequency of a term
cf	collection frequency of a term
V	vocabulary of collection (no. of unique terms)
C	size of collection (total number of terms)
0.5	<i>the constant 0.5</i>
1	<i>the constant 1</i>
10	<i>the constant 10</i>
Parameters	7 runs of Population 100 for 50 generations

Table 2. Term-Frequency Weighting Problem Terminals

Terminal	Description
tf	raw term-frequency of a term
1	<i>the constant 1</i>
10	<i>the constant 10</i>
0.5	<i>the constant 0.5</i>
Parameters	7 runs of Population 100 for 50 generations

Table 3. Normalisation Weighting Problem Terminals

Terminal	Description
l	document length (unique terms)
l_{avg}	average document length (unique terms)
l_{dev}	standard deviation of lengths (unique terms)
tl	total document length (all terms)
tl_{avg}	average total document length (all terms)
tl_{dev}	standard deviation of document lengths (all terms)
ql	query length (unique terms)
ql	query total length (all terms)
1	<i>the constant 1</i>
10	<i>the constant 10</i>
0.5	<i>the constant 0.5</i>
Parameters	7 runs of Population 200 for 25 generations

Table 4. Document Collections

Collection	#Docs	words/doc	#Topics	short	medium	long
LATIMES	131,896	251.7	301-350	2.4	9.9	29.9
FBIS	130,471	249.9	351-400	2.4	7.9	21.9
FT91-93	138,668	221.8	401-450	2.4	6.5	18.7
OH90-91	148,162	81.4	0-63	-	7.9	-

¹ University of Ireland, Galway. email: ronan.cummins@nuigalway.ie, colmor@it.nuigalway.ie

Table 5. % MAP of benchmark and evolved schemes on unseen test collections

		short			medium			long		
Collection	Topics	idf_{piv}	idf_{rsj}	gw_t	idf_{piv}	idf_{rsj}	gw_t	idf_{piv}	idf_{rsj}	gw_t
LATIMES	301-350 (m)	17.83	17.91	18.12	19.11	19.16	22.49	13.57	13.79	24.27
FBIS	351-400 (m)	11.19	11.24	11.72	10.30	10.41	15.68	06.76	06.97	13.32
FT91-93	401-450 (m)	21.69	21.69	21.79	27.38	28.15	27.86	23.11	23.13	28.28
OH90-91	0-63 (m)	-	-	-	21.68	21.72	25.69	-	-	-
Collection	Topics	$Piv_{s=0}$	$BM25_{b=0}$	$tf.gw_t$	$Piv_{s=0}$	$BM25_{b=0}$	$tf.gw_t$	$Piv_{s=0}$	$BM25_{b=0}$	$tf.gw_t$
LATIMES	301-350 (m)	20.95	24.75	24.89	13.80	20.55	24.38	10.94	13.98	25.87
FBIS	351-400 (m)	16.30	19.98	20.27	13.40	13.47	19.06	08.45	08.35	16.25
FT91-93	401-450 (m)	22.50	31.38	31.35	23.62	33.03	32.37	19.36	26.59	30.72
OH90-91	0-63 (m)	-	-	-	18.40	25.36	28.80	-	-	-
Collection	Topics	Piv	$BM25$	$ntf.gw_t$	Piv	$BM25$	$ntf.gw_t$	Piv	$BM25$	$ntf.gw_t$
LATIMES	301-350 (m)	24.26	24.17	23.87	25.48	25.61	28.64	25.79	26.77	30.80
FBIS	351-400 (m)	15.90	17.55	19.89	17.92	19.53	24.26	17.59	20.03	24.21
FT91-93	401-450 (m)	30.38	31.27	33.98	34.47	35.33	36.57	34.49	35.35	36.86
OH90-91	0-63 (m)	-	-	-	26.76	28.08	29.84	-	-	-

for the term-frequency influence problem consisted of 32,000 document from the LATIMES collection and 37 medium topics. The training set for the normalisation problem consisted of the same 32,000 documents from the LATIMES collection but we used 12 short, 13 medium and 12 long topics for this problem as it has been suggested that query length may have an impact on normalisation. We tested the solutions for generality on collections from TREC disks 4 and 5 to test our schemes. Table 4 details the collections and lengths of short (title), medium (title and description) and long (title and description and narrative) queries. Standard stop-words are removed and remaining words are stemmed.

2.2 Benchmark Term-Weighting

The full BM25 and pivoted normalisation scheme with default values are used as benchmarks for the entire schemes. The default term-frequency influence value of $k_1 = 1.2$ and normalisation influence value of $b = 0.75$ is used for BM25 while the slope (s) set to 0.2 is used for the pivoted normalisation scheme (Piv). We use the BM25 ($BM25_{b=0}$) and pivoted normalisation scheme ($Piv_{s=0}$) with no normalisation (i.e. assuming all documents are of equal length) as benchmarks for the global schemes combined a term-frequency influence factor. We use the idf scheme as found in the BM25 (idf_{rsj}) and pivoted normalisation scheme (idf_{piv}) as benchmarks for the global part of the scheme. We use the actual within-query term-frequency scheme (qtf) with all schemes as in (1).

2.3 Term-Weighting Scheme

One of the best evolved global schemes is as follows:

$$gw_t = \frac{cf^2\sqrt{cf}}{df^3} \quad (2)$$

The best evolved term-frequency factor, based on the global scheme, is as follows:

$$\log\left(\frac{10}{\sqrt{(0.5/tf) + 0.5}}\right) = \log\left(\sqrt{\frac{200.tf}{1+tf}}\right) \quad (3)$$

We assume the term-frequency factor is normalised (ntf) as follows:

$$ntf = \log\left(\sqrt{\frac{200.\frac{tf}{n}}{1+\frac{tf}{n}}}\right) \quad (4)$$

where n is some normalisation factor. One of the best evolved normalisation schemes is as follows:

$$n = \sqrt{\log(qtl)} \times \log(qtl) \times \frac{l}{l_{avg}} \quad (5)$$

Queries of length one were given the same normalisation as queries of length two during testing as $n = 0$ when $qtl = 1$. This occurred as there was no query of length one in the training set for the normalisation problem.

3 Discussion and Conclusions

It is worth noting that none of the randomly created solutions were as good as the best solution from the final generation. We can see for the global weighting problem that the evolved solution presented has a higher MAP on all topic lengths and collections. The increase over the idf type schemes is quite large for medium and long queries. For the term-frequency influence problem (assuming no normalisation), we can see that the MAP of the evolved solution ($tf.gw_t$) is higher than the benchmarks on most collections at this point especially for longer queries. We can see that the term-frequency part of the Piv scheme is a lot poorer than the default term-influence setting for BM25 at this stage. We can see that normalisation is beneficial to all schemes for medium and long queries but slightly degrades some short queries. We can see that our full evolved scheme is the best performing scheme on the collections for all but the short queries on the LATIMES collection. We have shown that term-weighting schemes can be found by evolutionary techniques that fit certain known aspects of weighting schemes and also contain new features such as query length.

REFERENCES

- [1] Ronan Cummins and Colm O'Riordan, 'Evolving general term-weighting schemes for information retrieval: Tests on larger collections.', *Artif. Intell. Rev.*, **24**(3-4), 277–299, (2005).
- [2] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992.
- [3] S. E. Robertson and S. Walker, 'On relevance weights with little relevance information', in *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 16–24. ACM Press, (1997).
- [4] A. Singhal, 'Modern information retrieval: A brief overview', *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, **24**(4), 35–43, (2001).
- [5] Andrew Trotman, 'Learning to rank', *Information Retrieval*, **8**, 359 – 381, (2005).

Adaptation Knowledge Discovery from a Case Base

M. d'Aquin¹ and F. Badra¹ and S. Lafrogne¹ and J. Lieber¹ and A. Napoli¹ and L. Szathmary¹

Abstract. In case-based reasoning, the adaptation step depends in general on domain-dependent knowledge, which motivates studies on adaptation knowledge acquisition (AKA). CABAMAKA is an AKA system based on principles of knowledge discovery from databases. This system explores the variations within the case base to elicit adaptation knowledge. It has been successfully tested in an application of case-based decision support to breast cancer treatment.

1 INTRODUCTION

Case-based reasoning (CBR [4]) aims at solving a target problem thanks to a case base. A case represents a previously solved problem. A CBR system selects a case from the case base and then adapts the associated solution, requiring domain-dependent knowledge for adaptation. The goal of adaptation knowledge acquisition (AKA) is to extract this knowledge. The system CABAMAKA applies principles of knowledge discovery from databases (KDD) to AKA. The originality of CABAMAKA lies essentially in the approach to AKA that uses a powerful learning technique that is guided by a domain expert, according to the spirit of KDD. This paper proposes an original and working approach to AKA, based on KDD techniques.

CBR and adaptation. A case in a given CBR application is usually represented by a pair $(pb, Sol(pb))$ where pb represents a problem statement and $Sol(pb)$, a solution of pb . CBR relies on the *source cases* $(srce, Sol(srce))$ that constitute the *case base* CB. In a particular CBR session, the problem to be solved is called *target problem*, denoted by tgt . A case-based inference associates to tgt a solution $Sol(tgt)$, with respect to the case base CB and to additional knowledge bases, in particular \mathcal{O} , the *domain ontology* that usually introduces the concepts and terms used to represent the cases.

A classical decomposition of CBR consists in the steps of retrieval and adaptation. *Retrieval* selects $(srce, Sol(srce)) \in CB$ such that $srce$ is judged to be similar to tgt . The goal of adaptation is to solve tgt by modifying $Sol(srce)$ accordingly.

The work presented hereafter is based on the following model of adaptation, similar to *transformational analogy* [1]:

- ① $(srce, tgt) \mapsto \Delta pb$, where Δpb encodes the similarities and dissimilarities of the problems $srce$ and tgt .
- ② $(\Delta pb, AK) \mapsto \Delta sol$, where AK is the adaptation knowledge and where Δsol encodes the similarities and dissimilarities of $Sol(srce)$ and the forthcoming $Sol(tgt)$.
- ③ $(Sol(srce), \Delta sol) \mapsto Sol(tgt)$, $Sol(srce)$ is modified into $Sol(tgt)$ according to Δsol .

Adaptation is generally supposed to be domain-dependent in the sense that it relies on domain-specific adaptation knowledge. There-

fore, this knowledge has to be acquired. This is the purpose of *adaptation knowledge acquisition* (AKA).

A related work in AKA. The idea of the research presented in [3] is to exploit the variations between source cases to learn adaptation rules. These rules compute variations on solutions from variations on problems. More precisely, ordered pairs $(srce-case_1, srce-case_2)$ of similar source cases are formed. Then, for each of these pairs, the variations between the problems $srce_1$ and $srce_2$ and the solutions $Sol(srce_1)$ and $Sol(srce_2)$ are represented (Δpb and Δsol). Finally, the adaptation rules are learned, using as training set the set of the input-output pairs $(\Delta pb, \Delta sol)$. The experiments have shown that the CBR system using the adaptation knowledge acquired from the automatic system of AKA shows a better performance compared to the CBR system working without adaptation. This research has strongly influenced our work that is globally based on similar ideas.

2 CABAMAKA

Principles. CABAMAKA deals with *case base mining* for AKA. Although the main ideas underlying CABAMAKA are shared with those presented in [3], the followings are original ones. The adaptation knowledge that is mined has to be validated by experts and has to be associated with explanations that make it understandable by the user. In this way, CABAMAKA may be considered as a semi-automated (or interactive) learning system. Another difference with [3] lies in the volume of the cases that are examined: given a case base CB where $|CB| = n$, the CABAMAKA system takes into account every ordered pair $(srce-case_1, srce-case_2)$ with $srce-case_1 \neq srce-case_2$ (whereas in [3], only the pairs of *similar* source cases are considered, according to a fixed criterion). Thus, the CABAMAKA system has to cope with $n(n - 1)$ pairs, a rather large number of elements, since in our application $n \simeq 750$. $(n(n - 1) \simeq 5 \cdot 10^5)$. This is why efficient techniques of knowledge discovery from databases (KDD [2]) have been chosen for this system.

Principles of KDD. The goal of KDD is to discover knowledge from databases, with the supervision of an analyst (expert of the domain). A KDD session usually relies on three main steps: data preparation, data-mining and interpretation.

Data preparation is based on formatting and filtering operations. The formatting operations transform the data into a form allowing the application of the chosen data-mining operations. The filtering operations are used for removing noisy data and for focusing the data-mining operation on special subsets of objects and/or attributes.

Data-mining methods are applied to extract pieces of information from the data. These pieces of information have some regular

¹ LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy 2-UHP), BP 239, 54506 Vandœuvre-lès-Nancy, FRANCE

properties allowing their extraction. For example, CHARM [5] is a data-mining algorithm that performs efficiently the extraction of *frequent closed itemsets (FCIs)*. CHARM inputs a database in the form of a set of transactions, each *transaction T* being a set of boolean properties or *items*. An *itemset I* is a set of items. The support of *I*, $\text{support}(I)$, is the proportion of transactions *T* of the database possessing *I* ($I \subseteq T$). *I* is frequent, with respect to a threshold $\sigma \in [0; 1]$, whenever $\text{support}(I) \geq \sigma$. *I* is closed if it has no proper superset *J* ($I \subsetneq J$) with the same support.

Interpretation aims at interpreting the output of data-mining i.e. the FCIs in the present case, with the help of an analyst. In this way, the interpretation step produces new knowledge units (e.g. rules).

Formatting. The formatting step of CABAMAKA inputs the case base CB and outputs a set of transactions obtained from the pairs $(\text{srce-case}_1, \text{srce-case}_2)$. It is composed of two substeps. During the first substep, each $\text{srce-case} = (\text{srce}, \text{Sol}(\text{srce})) \in \text{CB}$ is formatted in two sets of boolean properties: $\Phi(\text{srce})$ and $\Phi(\text{Sol}(\text{srce}))$. The computation of $\Phi(\text{srce})$ consists in translating srce from the problem representation formalism to $2^{\mathcal{P}}$, \mathcal{P} being a set of boolean properties. Possibly, some information may be lost during this translation, but this loss has to be minimized. Now, this translation formats an expression srce expressed in the framework of the domain ontology \mathcal{O} to an expression $\Phi(\text{srce})$ that will be manipulated as data, i.e. without the use of a reasoning process. Therefore, in order to minimize the translation loss, it is assumed that if $p \in \Phi(\text{srce})$ and p entails q (given \mathcal{O}) then $q \in \Phi(\text{srce})$. In other words, $\Phi(\text{srce})$ is assumed to be deductively closed given \mathcal{O} in the set \mathcal{P} . The same assumption is made for $\Phi(\text{Sol}(\text{srce}))$. How this first substep of formatting is computed in practice depends heavily on the representation formalism of the cases.

The second substep of formatting produces a transaction $T = \Phi((\text{srce-case}_1, \text{srce-case}_2))$ for each ordered pair of distinct source cases, based on the sets of items $\Phi(\text{srce}_1)$, $\Phi(\text{srce}_2)$, $\Phi(\text{Sol}(\text{srce}_1))$ and $\Phi(\text{Sol}(\text{srce}_2))$. Following the model of adaptation presented in introduction (items ①, ② and ③), T has to encode the properties of Δ_{pb} and Δ_{sol} . Δ_{pb} encodes the similarities and dissimilarities of srce_1 and srce_2 , i.e.:

- The properties common to srce_1 and srce_2 (marked by “=”),
- The properties of srce_1 that srce_2 does not share (“-”) and
- The properties of srce_2 that srce_1 does not share (“+”).

All these properties are related to problems and thus are marked by pb . Δ_{sol} is computed in a similar way and $\Phi(T) = \Delta_{\text{pb}} \cup \Delta_{\text{sol}}$. For example,

$$\begin{aligned} \text{if } & \begin{cases} \Phi(\text{srce}_1) = \{a, b, c\} & \Phi(\text{Sol}(\text{srce}_1)) = \{A, B\} \\ \Phi(\text{srce}_2) = \{b, c, d\} & \Phi(\text{Sol}(\text{srce}_2)) = \{B, C\} \end{cases} \\ \text{then } & T = \{a_{\text{pb}}, b_{\text{pb}}, c_{\text{pb}}, d_{\text{pb}}, A_{\text{sol}}, B_{\text{sol}}, C_{\text{sol}}\} \end{aligned} \quad (1)$$

Mining. The extraction of FCIs is computed thanks to CHARM (in fact, thanks to a tool based on a CHARM-like algorithm) from the set of transactions. A transaction $T = \Phi((\text{srce-case}_1, \text{srce-case}_2))$ encodes a specific adaptation $((\text{srce}_1, \text{Sol}(\text{srce}_1)), \text{srce}_2) \mapsto \text{Sol}(\text{srce}_2)$. An FCI extracted may be considered as a generalization of a set of transactions. For example, if $I_{\text{ex}} = \{a_{\text{pb}}, c_{\text{pb}}, d_{\text{pb}}, A_{\text{sol}}, B_{\text{sol}}, C_{\text{sol}}\}$ is an FCI, I_{ex} is a generalization of a subset of the transactions including the transaction T of equation (1): $I_{\text{ex}} \subseteq T$. The interpretation of this FCI as an adaptation rule is explained below.

Interpretation. The interpretation step is supervised by the analyst. The CABAMAKA system provides the analyst with the extracted FCIs and facilities for navigating among them. The analyst may select an FCI, say I , and interpret I as an adaptation rule. For example, the FCI I_{ex} may be interpreted in the following terms:

if a is a property of srce but is not a property of tgt ,
 c is a property of both srce and tgt ,
 d is not a property of srce but is a property of tgt ,
 A and B are properties of $\text{Sol}(\text{srce})$ and
 C is not a property of $\text{Sol}(\text{srce})$
then the properties of $\text{Sol}(\text{tgt})$ are
 $\Phi(\text{Sol}(\text{tgt})) = (\Phi(\text{Sol}(\text{srce})) \setminus \{A\}) \cup \{C\}$.

This has to be translated as an adaptation rule r of the CBR system. Then the analyst corrects r and associates an explanation with it.

Implementation. The application domain of the CBR system we are developing is breast cancer treatment: in this application, a problem pb describes a class of patients with a set of attributes and associated constraints (holding on the age of the patient, the size and the localization of the tumor, etc.). A solution $\text{Sol}(\text{pb})$ of pb is a set of therapeutic decisions (in surgery, chemotherapy, etc.). The requested behavior of the CBR system is to provide a treatment and explanations on this treatment proposal. This is why the analyst is required to associate an explanation to a discovered adaptation rule.

The problems, solutions and the domain ontology of the application are represented in OWL DL (recommendation of the W3C).

3 CONCLUSION

The CABAMAKA system presented in this paper is inspired by the research presented in [3] and by the principles of KDD for the purpose of semi-automatic adaptation knowledge discovery. It has enabled to discover several useful adaptation rules for a medical CBR application. It has been designed to be reusable for other CBR applications: only a few modules of CABAMAKA are dependent on the formalism of the cases and of the domain ontology, and this formalism, OWL DL, is a well-known standard. One element of future work consists in searching for ways of simplifying the presentation of the numerous extracted FCIs to the analyst. This involves an organization of these FCIs for the purpose of navigation among them. Such an organization can be a hierarchy of FCIs according to their specificities or a clustering of the FCIs in themes.

REFERENCES

- [1] J. G. Carbonell, ‘Learning by analogy: Formulating and generalizing plans from past experience’, in *Machine Learning, An Artificial Intelligence Approach*, ed., R. S. Michalski and J. G. Carbonell and T. M. Mitchell, chapter 5, 137–161, Morgan Kaufmann, Inc., (1983).
- [2] M.H. Dunham, *Data Mining – Introductory and Advanced Topics*, Prentice Hall, Upper Saddle River, NJ, 2003.
- [3] K. Hanney and M. T. Keane, ‘Learning Adaptation Rules From a Case-Base’, in *Advances in Case-Based Reasoning – Third European Workshop, EWCBR’96*, eds., I. Smith and B. Faltings, LNAI 1168, pp. 179–192. Springer Verlag, Berlin, (1996).
- [4] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.
- [5] M. J. Zaki and C.-J. Hsiao, ‘CHARM: An Efficient Algorithm for Closed Itemset Mining’, in *SIAM International Conference on Data Mining SDM’02*, pp. 33–43, (Apr 2002).

Polynomial Conditional Random Fields for signal processing

Trinh-Minh-Tri Do and Thierry Artières¹

Abstract. We describe Polynomial Conditional Random Fields for signal processing tasks. It is a hybrid model that combines the ability of Polynomial Hidden Markov models for modeling complex dynamic signals and the discriminant power of Conditional Random Fields. We detail the learning of these models and report experimental results on handwriting recognition.¹

1 INTRODUCTION

Classification, segmentation and labelling of sequential data are major problems in many application fields such as bioinformatics, signal processing (e.g. handwriting), information extraction and so on. Generally speaking these tasks consist in transforming an observed input signal or sequence into a sequence of labels. For decades, Hidden Markov Models (HMMs) have been the most popular approach although they rely on strong independence assumptions and despite they are learned using a non discriminant criterion, namely Maximum Likelihood Estimation. This comes from the fact that HMMs are generative models and that they define a joint probability distribution on the sequence of observations X and the associated label sequence Y .

A number of models were proposed for relaxing classical independence assumptions; most rely on segmental HMMs where trajectory models (e.g. polynomial) are used in each state to implement probability density functions at the segment level [1]. Besides conditional models were recently proposed for sequence processing tasks, Conditional Random Field (CRF) is the most popular one [2]. CRFs aim at modelling directly the conditional distribution $P(Y|X)$, they are attractive since first they are learned with a discriminant criterion where each label sequence is viewed as a possible class for an input sequence and second, being conditional, these models do not explicitly model input signals distribution so that they do not require any simplifying assumptions over X . However, CRF rely on the definition by hand of set of (eventually many) features, computed from observations X and label sequence Y . In a way underlying assumptions over X are implicit and hidden in the manual design of features. Up to now CRFs have been used in specific text processing tasks such as information extraction, FAQ segmentation, POS-tagging for which efficient features may be easily designed. The use of conditional models for continuously varying signals is less straightforward. This paper aims at developing a model which combines the accurate modelling of polynomial HMMs (PHMMs) with the efficiency of CRF discriminant learning. We first present Polynomial CRF (PCRF) and then we report comparative experimental results on on-line digit recognition.

¹ Pierre and Marie Curie University, France, email: Trinh-Minh-Tri.Do@lip6.fr, Thierry.Artieres@lip6.fr

2 POLYNOMIAL CRF

Let $X = (x_1, x_2, \dots, x_T)$ with $x_t \in \mathbb{R}^d$ be an input signal. Assuming that successive observations are generated according to a polynomial function of time, this sequence of observations may be modelled with a polynomial regression function:

$$x_t = f(t) = \sum_{k=0}^P a_k f_k(t) + \varepsilon(t) \quad (1)$$

where P is the degree of f , a_k are coefficients of polynomial functions f_k and $\varepsilon(t)$ is an i.i.d. residual. $\{f_k\}_k$ consists of a fixed set of polynomials (e.g. x^k). Assuming $\varepsilon(t)$ is a white Gaussian noise, the likelihood of a sequence may be computed as:

$$P(X) \propto e^{-\frac{1}{2} \sum_{t=1}^T \left\| x_t - \sum_{k=0}^P a_k f_k(t) \right\|^2} \quad (2)$$

Such polynomial functions have been embedded in HMM to implement more accurate probability density functions (*pdf*) than traditional Gaussian distributions. These Polynomial HMMs (PHMMs) have been shown to be more accurate in speech modelling and recognition tasks, they are trained using a Maximum Likelihood training algorithm which is a generalization of the standard HMM training algorithm [1].

Besides in CRFs the conditional probability of the output Y (i.e. the label sequence) given the input X (i.e. the sequence of observations) is defined with:

$$P(Y|X) = \frac{e^{W^T F(X, Y)}}{Z(X)} \quad (3)$$

where $F(X, Y)$ is a feature vector, W is the weight vector, $W^T F(X, Y)$ is their dot product, and $Z(X)$ is a normalization factor. Training consists in maximizing the conditional log likelihood over the training set w.r.t. W :

$$L = \sum_i \log P(Y_i | X_i) \quad (4)$$

Our model is an extension of CRFs for signal classification using segmental (polynomial) features, i.e. features that are defined on segments of successive observations. In this model, we use an ideal polynomial curve describing the sequence of successive observations in each state. Let S be a segmentation of X in the model, i.e. a state sequence. It may be decomposed into a sequence

of segments where a segment stands for a sequence of successive observations that are assigned to the same state. Let note s_j the j^{th} segment and let β_j stands for the starting time of s_j . Then $s_j = (x_{\beta_j}, \dots, x_{\beta_{j+1}-1})$. The probability $P(Y/X)$ may then be defined as:

$$P(Y/X) = \frac{1}{Z(X)} \sum_S e^{\text{score}(S, Y, X)} \quad (5)$$

where $\text{score}(S, Y, X)$ stands for the score of the label sequence Y along a particular segmentation S for input X . It may be further decomposed into:

$$\text{score}(S, Y, X) = \sum_{j=1}^{|S|} \text{score}(s_j, q_{j,Y}, X) \quad (6)$$

Where $q_{j,Y}$ is the state corresponding to the j^{th} segment in segmentation Y and $\text{score}(s_j, q, X)$ stands for the score associated to j^{th} segment and state q . Following works on polynomial regression, this score is linked to how well the segment fits with the trajectory model associated to the state. Following works in [1] and using Eq. (1) and Eq. (2), it is computed as:

$$\text{score}(s_j, q, X) = \sum_{t=\beta_j}^{\beta_{j+1}-1} \left\| x_t - \sum_{k=0}^P a_k^q f_k(t - \beta_j) \right\|^2 \quad (7)$$

Where a_k^q stand for the coefficients of the polynomial function in state q . To perform sequence classification we build a CRF whose architecture is illustrated in Fig. 1. There is one left-right chain model for each class. At test time, the most probable sequence of states (i.e. left-right chain) indicates the recognized class [3].

Training aims at estimating model's parameters, i.e. a_k^q . As for standard CRFs, this is done through maximizing the criterion in Eq. (4). There are two main problems for training polynomial CRF. First, unlike what is usually done with standard CRFs, the training database is only partially labelled. This means that the segmentation S corresponding to a training sequence is not known, only its class is known. This problem may be solved by introducing, in the training algorithm, hidden variables that correspond to segmentations S of the training sequences [3,4]. One can then use an EM-like learning algorithm consists then in iterating two steps. In the E-step, we one estimates the distribution over hidden variables $P(S/X, Y)$. In the M-step one updates the model parameters to reach a better likelihood [3]. Second, unlike standard CRFs, training involves a non linear optimization problem since the usual dot product in Eq. (3) is replaced here by a non linear term (Eq. (7)). Hence, in the M-step algorithm, parameters a_k^q are updated using a standard gradient descent method till convergence.

3 EXPERIMENTS

We carried out experiments on on-line digit recognition. On-line handwriting signals are captured with an electronic pen or a digitalized tablet; there are temporal sequences of pen coordinates. As is usually done in handwriting processing, raw signals are first normalized and transformed into a sequence of direction feature (angle of the trajectory with the horizontal axis). There are about 50 samples for every digit '0' to '9'. We report in Table 1 comparative results of a generative system (PHMMs) and our discriminative system (PCRFs) that have been obtained using cross validation. Performances are reported for various settings concerning the number of states in the models (1 to 4) and the

Table 1. Accuracy of PHMM and PCRF for digit recognition as a function of the number of states per digit model and of the polynomial degree.

	#states	P=0	P=1	P=2	P=3
PHMMs	1	67.4	88.2	93.0	93.6
	2	92.2	97.2	98.0	98.4
	3	96.4	98.6	98.6	98.6
	4	96.6	98.6	98.4	98.6
PCRFs	1	65.4	89.8	96.8	97
	2	98.0	99.0	99.0	98.8
	3	99.2	99.8	99.4	99.2
	4	98.6	98.8	98.8	98.2

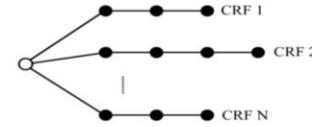


Figure 1. Mixture of chained CRF (i.e. with a chain structure) for sequence classification. Each chain corresponds to a class.



Figure 2. Mean samples of digits learned by a generative system (PHMM) on the left and a discriminant system (PCRF) on the right. Both systems are based on two-states models.

degree of polynomials (0 to 3). Almost whatever the number of states and the degree of polynomials PCRFs significantly outperform PHMMs. Although increasing the number of states in PHMMs improves accuracy it does not easily reach performance of PCRFs. In order to investigate deeper the behaviour of the models we generated from the learned models "mean samples" using polynomial trajectories learned in each state. Figure 2 compares these samples for the generative PHMM system as well as for the discriminant PCRF. One may see that although PHMM perform much lower, they allow modelling accurately the input signals. On the contrary PCRF, being discriminant, have focused on what distinguishes input signals of the ten digits so that mean samples do not represent well the original digits.

4 CONCLUSION

We presented Polynomial Conditional Random Fields for signal processing and discussed their learning. These models are hybrid and allow discriminant learning of trajectory models for signal classification and segmentation. PCRFs are shown to outperform non discriminant comparable polynomial generative models.

5 REFERENCES

- [1] Deng L., Aksmanovic M., Sun X. and Wu C.F.J, *Speech Recognition Using Hidden Markov Models with Polynomial Regression Functions as Nonstationary States*, IEEE Transactions on Speech and Audio Processing, 2:4, 507-520 (1994).
- [2] Lafferty J., McCallum A. and Pereira F., *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, International Conf. on Machine Learning, 282-289. Morgan Kaufmann, San Francisco, CA (2001).
- [3] Do T.-M.-T., Artières T., *Conditional Random Field for tracking user behavior based on his eye's movements*, NIPS'05 Workshop on Machine Learning for Implicit Feedback and User Modeling (2005).
- [4] Sarawagi S. and Cohen W., *Semi-Markov Conditional Random Fields for Information Extraction*, Advances in Neural Information Processing Systems NIPS (2004).

Stream Clustering Based on Kernel Density Estimation

Stefano Lodi and Gianluca Moro and Claudio Sartori¹

Abstract. We present a novel algorithm for clustering streams of multidimensional points based on kernel density estimates of the data. The algorithm requires only one pass over each data point and a constant amount of space, which depends only on the accuracy of clustering. The algorithm recognizes clusters of nonspherical shapes and handles both inserted and deleted objects in the input stream. Querying the membership of a point in a cluster can be answered in constant time.

1 Introduction

In many emerging scenarios, applications process the output of high speed, high volume data sources that originate continuous, unbounded flows of data, which have been termed *data streams*. Examples of data streams are call records of telephone companies, click streams, environmental data, and data from sensor networks. Stream processing is subject to tight constraints: Huge volume and high speed make random accesses and multiple scans of a memory image of the entire stream clearly infeasible, whereas unboundedness requires algorithms with constant time complexity per update, to avoid continuously increasing response times.

The design of stream data mining algorithms is especially demanding, as traditional solutions often entail combinatorial computations or exploring mutual relationships among the data. A fundamental data mining problem which has recently been studied in the stream domain is *data clustering*. Most formal definitions of the clustering problem capture the intuitive idea that similar objects must be grouped into the same cluster, whereas dissimilar objects must belong to different clusters. All approaches avoid storing the data records and maintain various kinds of statistics on the data, mostly in constant time. The statistics are sufficient to describe the clustering structure of the stream, and to answer cluster membership queries, at the price of some degree of inaccuracy.

We propose a novel stream clustering algorithm based on nonparametric kernel density estimation, capable of recognizing nonspherical clusters and allowing for both insertions and deletions of points from the stream.

Nonparametric density estimation is the construction of an estimate of the probability density function without assuming the data are drawn from a parametric family of distributions [5]. *Kernel estimators*, or *Parzen* estimators, are a popular family of nonparametric estimators [4]. Kernel estimators are expressed as summations of scaled copies of a single function; each copy is shifted and centered at a data point. Let us assume a set $D = \{\vec{x}_i \mid i = 1, \dots, N\} \subseteq \mathbb{R}^d$ of multidimensional data points. Kernel estimators formalize the idea that every data point contributes to the value of the estimate at a space

vector $\vec{x} \in \mathbb{R}^d$ an amount that monotonically decreases with distance from \vec{x} . Let $K: \mathbb{R} \rightarrow \mathbb{R}_+ \cup \{0\}$ be a non-negative, non-increasing function with unit integral on \mathbb{R} . K is called a *kernel function*. Examples of kernel functions are the square pulse function $\frac{1}{4}(\text{sign}(x+1) - \text{sign}(x-1))$, and the Gaussian function $\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$. A *kernel density estimate* $\hat{\phi}_{K,h}[D](\vec{x}) : \mathbb{R}^d \rightarrow \mathbb{R}_+ \cup \{0\}$ is defined as the sum over all data records \vec{x}_i in D of the differences between \vec{x} and \vec{x}_i , scaled by a factor h , called *window width*, and weighted by the kernel function K :

$$\hat{\phi}_{K,h}[D](\vec{x}) = \frac{1}{Nh^d} \sum_{i=1}^N K\left(\frac{1}{h}(\vec{x} - \vec{x}_i)\right). \quad (1)$$

The estimate is therefore a sum of bumps centered at the data points, and the flatness of the bumps is controlled by h . Thus, the smoothness of the estimate depends on the window width h .

Nonparametric density estimates have been used in several data clustering algorithms both in the statistical literature [5] and in the data mining literature [2, 3]. Clustering based on nonparametric estimates offers numerous advantages: immunity to noise, recognition of nonspherical clusters, and data-driven, objective criteria to optimally choose K and h for a given dataset. The intuition underlying all approaches is that well separated, high density regions around local maxima of the estimate correspond to clusters. In a *center-defined* cluster [2], all points are connected to a single local maximum having a sufficiently large density by an uphill path. In [5], a forest of points is constructed such that every path from a leaf to a root is an uphill path. All these approaches cannot be adapted easily to the stream environment since they do not maintain information allowing for quick, repeated updates of the clusters as new points arrive.

2 Clustering Data Streams

The approaches reviewed in the previous section build polygonal paths; the direction of every segment starting at \vec{x} approximates the direction of the gradient of $\hat{\phi}[D](\vec{x})$. In our approach, we relax this condition and allow for curves along which $\hat{\phi}[D](\vec{x})$ increases at any rate.

Definition 1. $C \subseteq D$ is a subcluster for a local maximizer \vec{x}^* of $\hat{\phi}[D]$ if and only if $\vec{x}_i \in C$ implies that there exists a simple continuous curve $\vec{c}: [0, 1] \rightarrow \mathbb{R}^d$ satisfying $\vec{c}(0) = \vec{x}_i$, $\vec{c}(1) = \vec{x}^*$, and $\hat{\phi}[D](\vec{c}(t_1)) > \hat{\phi}[D](\vec{c}(t_2))$ when $t_1 > t_2$. C is a cluster if C is a maximal subcluster for some local maximizer of $\hat{\phi}[D]$.

We now define our problem. Let S be a stream of d -dimensional points $\dots, \vec{x}^{(-1)}, \vec{x}^{(0)}, \vec{x}^{(1)}, \vec{x}^{(2)}, \dots$, with $\vec{x}^{(i)} \in \mathbb{R}^d$, for $i \in \mathbb{Z}$, and let the index $i_0 \in \mathbb{Z}$ of the first data object read from input be given. Further let Q be a stream of cluster membership queries $\dots, q[-1], q[0], q[1], q[2], \dots$, where each $q[j]$ asks whether two objects $\vec{x}^{(i)}$,

¹ Department of Electronics, Computer Science and Systems, University of Bologna, Viale Risorgimento 2, IT-40136 Bologna BO, Italy, email: {stefano.lodi,gianluca.moro,claudio.sartori}@unibo.it

$\vec{x}^{(i')}, i, i' \leq j$ are in the same cluster. The problem is to output correct results (w.r.t. Definition 1) to the queries Q in such a way that: the number of data objects read from the input stream S between reading of $q[j]$ and writing its result is $O(1)$; the number of computation steps executed between reading $\vec{x}^{(i)}$ and $\vec{x}^{(i+1)}$, $i \geq i_0$, is $O(1)$; the number of computation steps between reading $\vec{x}^{(i)}$ and the earliest time the result to a query q equals the result of q applied to the clustering of $\{\vec{x}^{(i_0)}, \vec{x}^{(i_0+1)}, \dots, \vec{x}^{(i)}\}$ is $O(1)$. These requirements specify that a stream algorithm must not be left behind by an increasingly large lag as time passes.

A constant bound on the number of items which have to be updated to yield an updated clustering is attainable only if we can avoid building and maintaining a forest over the data objects. Alternatively, we maintain a forest of directed grid trees the vertex set of which is a set of regularly spaced points of \mathbb{R}^d covering the support of the density estimate. Note that smooth kernel functions with bounded support exist; for example, take $K(x) = \exp(1/(x^2 - 1))$ if $|x| < 1$; 0 otherwise.

Each vertex is associated to the value the density estimate takes at the vertex, that is, to a sample of $\hat{\phi}$. Let w be a sampling frequency, and let us denote any element of \mathbb{Z}^d by \vec{n} . Let $\vec{n}_1/w, \vec{n}_2/w$ be the corners of the minimal d -dimensional rectangle containing the support of $\hat{\phi}$. Note that \vec{n}_1, \vec{n}_2 can always be set initially so to contain the bounding box of the data, which is always known in practical cases, extended by h on all dimensions. The vertex set is therefore the d -dimensional rectangular grid of corners $\vec{n}_1/w, \vec{n}_2/w$ and spacing $1/w$ in all dimensions $\{\vec{n}/w : \vec{n}_1 \leq \vec{n} \leq \vec{n}_2\}$, where $\vec{n} \leq \vec{n}'$ if and only if \vec{n} holds componentwise. The main invariant we wish to maintain is the following.

Invariant 1. $(\vec{n}'/w, \vec{n}''/w)$ is an edge of the grid forest if and only if $\hat{\phi}[S](\vec{n}'/w) < \hat{\phi}[S](\vec{n}''/w)$ and \vec{n}'' maximizes $\hat{\phi}[S](\vec{n}/w)$ over all \vec{n}/w nearest to \vec{n}'/w , i.e., $\vec{n}'' = \text{argmax}_{\vec{n}} \{\hat{\phi}[S](\vec{n}/w) : \vec{n} = \vec{n}' + \vec{e}/w \text{ for some versor } \vec{e}\}$.

Therefore, in the data structures we store polygonal uphill paths constructed from segments that can only run parallel to some coordinate axis, and values of the density at every grid point. The structure can be updated as follows. When a new point $\vec{x}^{(i)}$ arrives from the stream S only the density at a bounded number of grid points can increase. (If deletions are allowed, their densities increase or decrease according to a sign attached to the new point.) The affected grid points are located inside the smallest hypercube containing the hypersphere of radius h , centered at $\vec{x}^{(i)}$. Edges going into or out of the hypercube must be checked for inversion. The number of density values and edge updates needed depends on h and w .

The forest induces a clustering of the space, for example by a nearest neighbour criterion. Therefore membership queries $q[i]$ in the stream Q can be answered by first finding the nearest vertex and then following the unique uphill path from the vertex. The cost is clearly independent of stream size, and depends only on the spacing w of the vertex set. Other types of queries can also be answered with similar costs. The total number of clusters can be maintained as the number of tree roots. Grid points which become roots, or are no longer roots can be detected easily during the updates. The size of a cluster can be computed by visiting the corresponding tree, and summing the values of a counter attached to every grid point. The counter is incremented whenever the grid point is the nearest grid point to a new stream point.

Preliminary experiments have shown that considerable accuracy can be achieved on clusters of arbitrary shape. The dataset shown in Figure 1 has been processed as a stream setting $w = 2$ and $w = 4$.

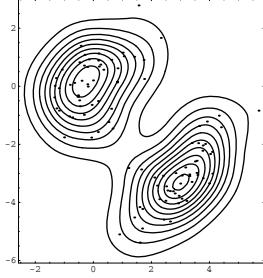


Figure 1. A test data set with a contour plot of its kernel estimate ($h = 0.7$).

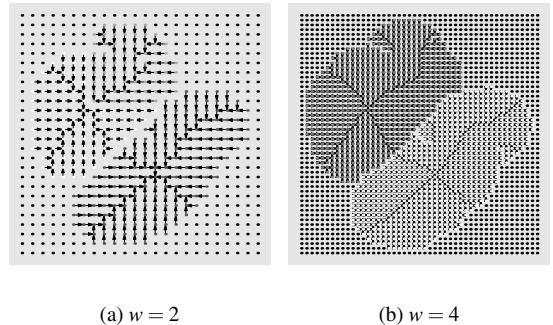


Figure 2. Grid forests constructed for $w = 2$ and $w = 4$.

The resulting grid forests are shown in Figures 2(a) and 2(b) respectively. Equal gray levels correspond to equal clusters. For $w = 2$, the top left cluster was erroneously split nearby its maximum since the directions of the grid edges do not yield an increase in $\hat{\phi}[S]$. Halving the spacing between grid points allows for correct recognition. This behaviour has been apparent in all experiments and, in fact, should reflect a formal convergence property showing that every path in the grid forest, although not guaranteed to be monotonic for $\hat{\phi}[S](\vec{x})$, as w increases, should converge to a monotonic path for $\hat{\phi}[S](\vec{x})$. The speed of convergence must also be investigated, as well as possible heuristics to merge clusters when w is large (e.g. due to limited resources). Moreover, the speed of processing stream data points must be experimentally evaluated and compared to the rate of a fast algorithm, e.g. CluStream [1]. Methods to choose h should also be investigated. In fact, the optimal value of h decreases with sample size (as $N^{-1/(d+4)}$) [5], therefore no value can be uniformly optimal.

REFERENCES

- [1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, ‘A framework for clustering evolving data streams’, in *VLDB 2003*, pp. 81–92, Berlin, Germany, (September 9–12 2003). Morgan Kaufmann.
- [2] A. Hinneburg and D. A. Keim, ‘An efficient approach to clustering in large multimedia databases with noise’, in *Proc. KDD-98*, pp. 58–65, New York City, New York, USA, (1998). AAAI Press.
- [3] M. Klusch, S. Lodi, and G. Moro, ‘Distributed clustering based on sampling local density estimates’, in *Proc. IJCAI-03*, pp. 485–490, Acapulco, Mexico, (August 2003). AAAI Press.
- [4] E. Parzen, ‘On estimation of a probability density function and mode’, *Ann. Math. Statist.*, **33**, 1065–1076, (1962).
- [5] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

Version space learning for possibilistic hypotheses

Henri Prade and Mathieu Serrurier¹

Abstract. In this paper, we are interested in learning stratified hypotheses from examples and counter-examples associated with weights that express their prototypical importance. It leads to an extension of the well-known version space learning framework. In order to do that, we emphasize that the treatment of positive and negative examples in version space learning is reminding of a bipolar revision process recently studied in the setting of possibilistic information representation. Bipolarity appears when the positive and negative sides of information are specified in a distinct way. Then, we use the possibilistic bipolar representation setting, which distinguishes between what is guaranteed to be possible, and what is simply not impossible, as a basis for extending version space learning to examples associated with possibility degrees. It allows us to define a formal framework for learning layered hypotheses.

1 Introduction

Version space learning [3] is a general framework for concept learning. It proposes to identify all the hypotheses coherent with a set of examples or counter examples. The way for treating positive and negative examples during the learning process reminds a bipolar revision process, that is at work when positive information is said apart from negative information [1]. Negative information states what is known to be impossible, what is rejected, while positive information states what is possible for sure, feasible or satisfactory. Examples and counter examples w. r. t. an if-then rule are instances of positive and negative information. When a bipolar view of information is maintained, consistency between positive and negative information, namely what is positively assessed is included in what is not negatively assessed, should be enforced when new information is received. We first point out the similarity between version space learning and bipolar revision, when learning from a set of examples and counter-examples. Then, examples are associated with possibility degrees that describe how much a positive example is guaranteed to be possible or how much a negative example is not impossible. By taking advantage of bipolar possibilistic revision, we propose an original algorithm for version space learning in order to identify the set of layered hypotheses that are sound w. r. to t. examples.

2 Version space and bipolar information

Let \mathcal{X} denote a *feature space*. In the bipolar [1] possibilistic setting, two $[0, 1]$ -valued possibility distributions over \mathcal{X} are considered : δ describes to what extent configurations are guaranteed to be possible and π describes what is not impossible. These two distributions are consistent in the bipolar framework iff $\forall x \in \mathcal{X}, \delta(x) \leq \pi(x)$. Given δ and π , which respectively describe what is guaranteed

to be possible and what is not impossible, when a new piece of information of the form π_{new} or δ_{new} is presented, a revision process takes place : $\forall x \in \mathcal{X} \pi_{revised}(x) = \max(\pi(x), \delta_{new}(x))$ and $\delta_{revised}(x) = \min(\pi_{new}(x), \delta(x))$. Let us describe the version space learning framework from the bipolar point of view. Let $\mathcal{U} = \{0, 1\}$ be a *concept space*, where 1 means that the example of the concept is positive (guaranteed possible i.e. $\delta(x) = 1$), and 0 means that the example is negative (totally impossible i.e. $\pi(x) = 0$). An hypothesis h is a mapping from \mathcal{X} to \mathcal{U} . \mathcal{H} denotes the hypotheses space. The version space framework takes as input a set $\mathcal{S} = \{(x_i, u_i)_{i=1, \dots, m} \text{ s.t } x_i \in \mathcal{X} \text{ and } u_i \in \mathcal{U}\}$ of m training examples, and a hypothesis space \mathcal{H} . The set \mathcal{H} is supposed to be equipped with a partial preorder \succeq expressing generality, formally: $h_1, h_2 \in \mathcal{H}, h_1 \succeq h_2$, iff $\{x \in \mathcal{X} | h_1(x) = 1\} \supseteq \{x \in \mathcal{X} | h_2(x) = 1\}$. An hypothesis h is *sound* w. r. t. a training example $(x, u) \in \mathcal{S}$ iff $h(x) = u$. If $\forall (x_i, u_i) \in \mathcal{S}, h$ is sound w.r.t (x_i, u_i) , then h is said to be sound w. r. t. \mathcal{S} . This framework defines the *version space* $\mathcal{V} = \{h \in \mathcal{H} | h \text{ is sound with } \mathcal{S}\}$. Version space learning aims at identifying the upper and the lower bounds of this version space \mathcal{V} . The upper bound \mathcal{V}_g will contain the most general hypotheses, i.e., the ones that classify more examples, whereas the lower bound \mathcal{V}_s will contain the most specific ones, i.e., the hypotheses that classify less examples, both being compatible with the training examples. The general algorithm for learning version space consists in updating \mathcal{V}_s and \mathcal{V}_g step by step by considering an increasing set of examples. If the example is positive, all the hypotheses of \mathcal{V}_g that are not coherent are removed, since they cannot be generalized, and all the hypotheses of \mathcal{V}_s that do not cover the examples are minimally generalized in order to cover it. On the contrary, if the example is negative the hypotheses that are incoherent with the example are removed if they are in \mathcal{V}_s or minimally specialized if they are in \mathcal{V}_g .

We can view a hypothesis as a possibility distribution, which states what feature configurations x are possible : $\forall h \in \mathcal{H}, \forall x \in \mathcal{X}, \mu_h(x) = h(x)$. Then, given a set of hypotheses, we define two possibility distributions : the most specific one $\forall H \in 2^{\mathcal{H}}, \forall x \in \mathcal{X}, \delta_H(x) = \min\{\mu_h(x); h \in H\}$ and the most general one $\forall H \in 2^{\mathcal{H}}, \forall x \in \mathcal{X}, \pi_H(x) = \max\{\mu_h(x); h \in H\}$. Let $\pi_{\mathcal{S}}$ and $\delta_{\mathcal{S}}$ be the possibility distributions that correspond respectively to the most general and the most specific distribution revised by the examples in \mathcal{S} . The set of hypotheses in \mathcal{V}_s contains the most specific hypotheses that cover all the positive examples and no negative examples. It means that these hypotheses must identify, if possible, only the situations that are *guaranteed to be possible* w. r. t. the set of examples and then $\delta_{\mathcal{S}} \leq \delta_{\mathcal{V}_s}$. In the same way, since \mathcal{V}_g contains the most general hypotheses that cover all the positive examples and no negative ones, these hypotheses describe the situations that are *not impossible* w. r. t. the set of examples and then $\pi_{\mathcal{S}} \leq \pi_{\mathcal{V}_g}$.

¹ IRIT - CNRS, 118 route de Narbonne, 31062 Toulouse Cedex 09, France
 email: prade,serrurier@irit.fr

3 Weighted extension of version space learning

Alg. 1 SpecializeStratifiedHypothesis

Require: $h_w = \{(h_1, \alpha_1), \dots, (h_n, \alpha_n)\}$ with $\alpha_1 \geq \dots \geq \alpha_n$

Require: $\langle x, 0, \alpha \rangle$

- 1: Let i such that $(h_i, \alpha_i) \in h_w$ $\alpha_i \geq \alpha$ and $\exists j$ such that $\alpha_i > \alpha_j \geq \alpha$
- 2: **for all** (h_j, α_j) such as $j > i$ **do**
- 3: **if** $h_j(x) = 1$ **then**
- 4: turn h_j in (h_j, α_j) into one of its minimal specialization
- 5: **if** $h_i(x) = 1$ **then**
- 6: turn (h_i, α_i) into (h'_i, α_i) where h'_i one of the minimal specialization of h_i
- 7: **if** $\alpha_i \neq \alpha$ **then**
- 8: add (h_i, α) in h_w

Alg. 2 GeneralizeStratifiedHypothesis

Require: $h_w = \{(h_1, \alpha_1), \dots, (h_n, \alpha_n)\}$ with $\alpha_1 \geq \dots \geq \alpha_n$

Require: $\langle x, 0, \alpha \rangle$

- 1: **if** $\exists i$ such that $(h_i, \alpha_i) \in h_w$ and $\alpha_i = \alpha$ **then**
- 2: return all h_w in which h_i is turned into one of its minimal generalization
- 3: **else**
- 4: Let i such that $(h_i, \alpha_i) \in h_w$ $\alpha_i > \alpha$ and $\exists j$ such that $\alpha_i > \alpha_j > \alpha$
- 5: add (h_i, α_i) in h_w such that h'_i is a minimal generalization of h_i

We are now interested in version space learning with examples associated with weights corresponding to possibility degrees in a linearly ordered scale, here $[0, 1]$ for simplicity. These possibility values have a different meaning according as the examples are positive or negative. If the example is positive, the weight corresponds to a guaranteed possibility degree. If the example is negative, the weight refers to the possibility distribution π that describes what is not impossible. The examples are then described by $\mathcal{S}_w = \{(x, u, \alpha)\}$ with $\alpha = \delta(x)$ if $u = 1$ and $\alpha = \pi(x)$ otherwise. The hypothesis we want to learn describes a possibility distribution that associates a possibility degree to each example, which corresponds to the degree of compatibility of this example w. r. t. the target concept. The hypothesis is then a mapping from \mathcal{X} to $[0, 1]$. Given an hypothesis space \mathcal{H} , we now consider a stratified set of classical hypotheses $h_w = (h_1, \alpha_1), \dots, (h_n, \alpha_n)$ with $\forall i, h_i \in \mathcal{H}$ and $\alpha_1 \geq \dots \geq \alpha_n$. The result of h_w for x is $\forall x \in \mathcal{X}, h_w(x) = \max_i \{\min(h_i(x), \alpha_i)\}$ and the possibility distribution induced by h_w is $\forall x \in \mathcal{X}, \mu_{h_w} = h_w(x)$. The generality ordering on stratified hypotheses corresponds to the partial pre-order on possibility distributions, and thus $h_w^1 \leq h_w^2$ iff $\mu_{h_w^1} \leq \mu_{h_w^2}$.

A stratified hypothesis is sound w. r. t. an example if it does not underestimate the possibility of a positive example and does not overestimate the possibility degree of a negative example. So, h_w is sound w. r. t. \mathcal{S}_w iff $\forall (x, u, \alpha) \in \mathcal{S}_w = h_w(x) \geq \alpha$ if $u = 1$ and $h_w(x) \leq \alpha$ if $u = 0$. A totally positive (guaranteed possible) example corresponds to $(x, 1, 1)$, and a totally negative (impossible) example to $(x, 0, 0)$. On the contrary, the examples of the form $(x, 1, 0)$ and $(x, 0, 1)$ have no influence on the learning algorithm according to the definition of soundness. Indeed, an example of the

form $(x, 1, 0)$ means just that x is not guaranteed to be a positive example and $(x, 0, 1)$ means that we are certain that it is possible for x to be a positive example. The algorithm for learning the version space in the case of bipolar continuous examples follows the same structure as the classical version space learning algorithm by using specialization and generalization operators described respectively in Alg. 1 and Alg. 2.

Proposition 1 We note $\delta_{\mathcal{S}_w}$ and $\pi_{\mathcal{S}_w}$ the possibility distributions obtained by revising the bipolar distribution ($\forall x \in \mathcal{X}, \delta_{\mathcal{S}_w}(x) = 0, \forall x \in \mathcal{X}, \pi_{\mathcal{S}_w}(x) = 1$) with the new information of the form $\forall (x, u, \alpha) \in S \pi_{new}(x) = \alpha$ if $u = 0$ or $\delta_{new}(x) = \alpha$ if $x = 1$. Given $\mathcal{V}_w = \langle \mathcal{V}_{sw}, \mathcal{V}_{gw} \rangle$ the continuous version space from the set of example S , we have $\forall x \in \mathcal{X}, \delta_{\mathcal{S}_w}(x) \leq \delta_{\mathcal{V}_{sw}}(x)$ and $\pi_{\mathcal{S}_w}(x) \geq \pi_{\mathcal{V}_{gw}}(x)$.

This proposition shows that we can use version space learning for describing bounds of possibility distributions according to a set of descriptions of what is not impossible or guaranteed to be possible. Since it is described in intention in the version space case, the bounds may be too restrictive w. r. t. the maximal bounds described by bipolar revision due to the limited description capabilities of the hypothesis language.

4 Related works and concluding remarks

In this paper, we have emphasized the similarities that exist between the version space learning framework and the bipolar binary revision process. According to this remark, we have proposed an extension of version space learning that deals with examples associated with possibility degrees. These degrees correspond to non impossibility degrees for negative examples and to guaranteed possibility degrees for positive examples. What is learned is the bounding of the set of stratified hypotheses that are coherent with the data. As for the classical setting, it is obvious that bipolar version space learning cannot be used directly in applications, but can be used as a formal framework for possibilistic concept learning.

This approach could then be applied to possibilistic inductive logic learning [4], which learns stratified hypotheses from weighted examples. In this context, it suggests to modify the treatment of examples in order to consider weights as bipolar information and then to adapt the specialization and generalization operators as described in this paper. This approach is clearly different from the fuzzy version space in [2]. In particular, our approach makes a difference for the meaning, and then for the treatment, of the levels associated to the examples according to their positive or negative nature, which is not the case in fuzzy version space. Indeed, in the fuzzy case, a version space is learnt for each *alpha*-cut of the training set. What is learnt is then a stratified sets of version spaces and the soundness of hypotheses w. r. t. examples is classical, while what is learnt here is a version space of stratified hypotheses.

REFERENCES

- [1] D. Dubois, H. Prade, and P. Smets, ‘Not impossible vs. guaranteed possible in fusion and revision’, in *Proc.of the 6th.European Conference (ESQARU 2001)*, pp. 522–531. Springer-Verlag, (2001).
- [2] E. Hüllermeier, ‘Inducing fuzzy concepts through extended version space learning’, in *Proc. of 10th Int. Fuzzy Systems Association (IFSA-03) - LNAI 2715*, pp. 677–684, (2003).
- [3] T. Mitchell, ‘Generalization as search’, *Artificial Intelligence*, **18**, 203–226, (1982).
- [4] M. Serrurier and H. Prade, ‘Possibilistic inductive logic programming’, in *Proc. ECSQARU’05 - LNAI 3571*, pp. 675–686, (2005).

Ensembles of Grafted Trees¹

Juan J. Rodríguez and Jesús M. Maudes²

Abstract. Grafted trees are trees that are constructed using two methods. The first method creates an initial tree, while the second method is used to complete the tree.

In this work, the first classifier is an unpruned tree from a 10% sample of the training data. Grafting is a method for constructing ensembles of decision trees, where each tree is a grafted tree. Grafting by itself is better than Bagging.

Moreover, grafted trees can also be used with any other ensemble method. It is clearly beneficial for Bagging and Random Forests. When using grafted trees with Boosting, the results depends of the considered variant. The best overall method is Grafted Random Forest.

1 INTRODUCTION

One of the current research areas in Machine Learning is the construction of ensembles of classifiers [8]. It is one of the easiest ways of trying to improve the accuracies of existing methods. Some ensemble methods work combining several classifiers obtained with the same method. Combining identical classifiers is not useful. Hence, it is necessary to devise some approach that allows the construction of different classifiers from the same method. This paper presents a new approach.

In Bagging [2], each classifier is constructed using a different data set. It is obtained using sampling with replacement. In the Random Subspace method [7] each classifier in the ensemble is constructed using a random subset of the features. The main idea of Boosting [11] is, when constructing a classifier, to give more importance to the examples that were more misclassified by the previous members of the ensemble.

Random Forests [3] is a variant of Bagging, where the classifiers are constructed using Random Trees. Another way of constructing Random Trees is presented in [5]. These methods are designed specifically for ensembles of decision trees. Ensembles of cascaded trees is another method designed specifically for decision trees [9]. Each decision tree is constructed using a different feature in the root. The rest of nodes are constructed as usual.

The term “grafting” has already appeared in the Machine Learning community [12]. In this reference more nodes are added to a tree in order to improve its accuracy. The main difference with the present work is that we are considering grafted trees as ensemble members, and consequently we are looking for diverse classifiers. Hence, our objective is that the initial decision trees were different.

¹ This work was partially supported by Spanish Ministry of Education and Culture, through grant DPI2005-08498 and Junta Castilla y León VA088A05.

² University of Burgos, Spain, email: {jjrodriguez,jmaudes}@ubu.es

2 GRAFTED CLASSIFIERS

Given two induction methods, \mathcal{M}_1 and \mathcal{M}_2 , a grafted classifier is constructed in the following way:

- Construct a classifier C_1 using the method \mathcal{M}_1 with the training data.
- For each different output, i , of the classifier C_1 , construct a classifier $C_{2,i}$ using the method \mathcal{M}_2 . The training data for this classifier is the subset of instances of the training data that are assigned to i by C_1 .

In order to classify instances, a grafted classifier first classifies the instance using C_1 . Let be i its output. The predicted class is the classification given by the classifier $C_{2,i}$.

Grafted trees and ensembles. If \mathcal{M}_1 and \mathcal{M}_2 are decision tree methods, the grafted classifiers will be called grafted trees. A grafted tree is a decision tree.

In the present paper we are going to consider only a method \mathcal{M}_1 . It constructs an unpruned decision tree from a random sample. The size of this sample is 10% of the size of the original data. For the method \mathcal{M}_2 we are going to consider pruned trees, unpruned trees and random trees. Given that these grafted trees have a random component, it is possible to obtain different trees using the same data set. Hence, a way to generate an ensemble of decision trees is constructing several grafted trees with the same data set. These ensembles are called Grafting ensembles.

These configurations for the methods \mathcal{M}_1 and \mathcal{M}_2 were selected because we wanted diverse and accurate classifiers. These grafted trees can be diverse because a small subset of the training data is used to construct the initial levels of the trees. The grafted trees can be as accurate as standard decision trees because none of the information in the training data (instances, attributes) is discarded.

Moreover, grafted trees can be used with any other ensemble method that can be used with decision trees. Hence, for each ensemble method, there will be a grafted variant of that method. For instance, Grafted-Bagging, Grafted-Boosting...

3 EXPERIMENTAL VALIDATION

Settings. For the experiments, WEKA [13] was used. The used decision tree methods (J48 —a reimplementation of C4.5— and Random Trees) and ensemble methods (Bagging, AdaBoost.M1, Random Forests) are from this library. Bagging and AdaBoost.M1 are used with J48. The method Random Forests is a special case of Bagging, when Random Trees are used as base classifiers.

We consider two variants of AdaBoost.M1: the reweighting version and the resampling version [6]. In the tables, they are denoted,

respectively, with (W) and (S). The number of classifiers in each ensemble was 25. For the experiments, 10 10-fold stratified cross validation was used.

For the comparison of results the *corrected resampled t-test statistic* from [10] was used (significance level: 5%). The 100 (10×10) results for each method and data set are considered as the input for the test.

The 35 considered data sets, from the UCI Repository [1], were: anneal, audiology, autos, balance-scale, breast-cancer, cleveland-14-heart, credit-rating, german-credit, glass, heart-statlog, hepatitis, horse-colic, hungarian-14-heart, hypothyroid, ionosphere, iris, kr-vs-kp, labor, letter, lymphography, mushroom, pima-diabetes, primary-tumor, segment, sick, sonar, soybean, splice, vehicle, vote, vowel-context, vowel-nocontext, waveform, wisconsin-breast-cancer and zoo.

Grafting ensembles. First, Grafting ensembles are compared to other ensemble methods. Table 1 shows the number of wins, draws and losses when comparing Grafting with another methods. The table also shows the number of wins, draws and losses according to the used statistical test. From this table we can conclude that:

- Grafting is better than Bagging.
- Grafting is worse than Random Forests according to the direct comparison of the accuracies. On the other hand, according to the used statistical test, Grafting is better than Random Forest 4 times and worse only 2 times.
- The comparison of the accuracies of Grafting and AdaBoost.M1 is very even when pruned trees are used. If the trees are unpruned, there is a slight advantage for AdaBoost.M1.

Combining grafting with other ensemble methods. Table 2 shows the comparison of the results between ensemble methods. For each ensemble method, the number of wins, draws and losses of the version with grafted trees against the version with standard trees is shown. The table also shows the number of wins, draws and losses according to the used statistical test. From this table it can be concluded that:

- Grafting is clearly beneficial for Bagging.
- If we only consider the used statistical test, there is no difference between using the standard Random Forest method and Grafted Random Forest. Nevertheless, in the direct comparison of the results there are 28 wins and only 6 losses.
- For boosting methods, grafting does not improve the results. For the reweight version of AdaBoost.M1, Grafting is slightly harmful. On the other hand, for the resample version of AdaBoost.M1, grafting is slightly beneficial.

Ranking the methods. Although it is customary to use paired tests for comparing several methods, they were not designed for that purpose [4]. A better way of ranking the methods is doing a ranking for each data set. For the data set, the first method has a rank of 1, the second 2 ... If there are ties, average ranks are assigned. For each method, its average rank is calculated. Table 3 shows this average ranking. It provide a fair comparison of the methods [4]. From this ranking we can conclude that:

- Grafted Random Forest is the best method. Random Forest without Grafting is rather far from the top positions.
- Bagging without Grafting is the worst method. Grafted Bagging works much better, specially with unpruned trees.

- The results for AdaBoost.M1 are rather mixed, because it is not clear if Grafting is beneficial or harmful. The best variant of AdaBoost.M1 is the version with resampling, grafted unpruned trees.
- Grafting by itself is only clearly better than Bagging.

Table 1. Comparison of grafting with other ensemble methods. The prefix (P- or U-) indicates if the trees were (P)runed or (U)npruned.

Method	Wins/Draws/Losses	Significant	
		Wins	Draws/Losses
P-Bagging	25/1/9	4/31/0	3/29/3
P-AdaBoost.M1 (W)	17/1/17	3/29/3	3/29/3
P-AdaBoost.M1 (S)	18/1/16	3/29/3	
U-Bagging	24/1/10	4/31/0	4/29/2
U-Random Forest	14/1/20	4/29/2	1/32/2
U-AdaBoost.M1 (W)	14/1/20	1/32/2	1/31/3
U-AdaBoost.M1 (S)	14/1/20	1/31/3	

Table 2. Comparison of the grafted variant of some ensembles methods with the standard version. The prefix (P- or U-) indicates if the trees were (P)runed or (U)npruned.

Method	Wins/Draws/Losses	Significant	
		Wins	Draws/Losses
P-Bagging	23/1/11	5/30/0	0/34/1
P-AdaBoost.M1 (W)	15/0/20	0/34/1	0/35/0
P-AdaBoost.M1 (S)	18/0/17	0/35/0	
U-Bagging	27/1/7	7/28/0	
U-Random Forest	28/1/6	1/33/1	
U-AdaBoost.M1 (W)	16/1/18	0/34/1	
U-AdaBoost.M1 (S)	20/1/14	1/34/0	

Table 3. Average ranks of the methods. The first prefix (G- or N-) indicates if Grafting was used or not. The second prefix (P- or U-) indicates if pruning was used.

Method	Average Ranking	Method	Average Ranking
G-U-Random Forest	6.14	N-U-AdaBoost.M1 (W)	8.27
G-U-AdaBoost.M1 (S)	7.64	N-U-Random Forest	8.31
G-U-Bagging	7.77	N-U-AdaBoost.M1 (S)	8.60
N-P-AdaBoost.M1 (W)	7.77	G-U-AdaBoost.M1 (W)	9.00
G-P-Grafting	8.00	G-P-AdaBoost.M1 (W)	9.14
G-P-AdaBoost.M1 (S)	8.06	G-U-Grafting	9.19
N-P-AdaBoost.M1 (S)	8.09	N-P-Bagging	10.64
G-P-Bagging	8.13	N-U-Bagging	11.24

REFERENCES

- [1] C. L. Blake and C. J. Merz, ‘UCI repository of machine learning databases’, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [2] L. Breiman, ‘Bagging predictors’, *Machine Learning*, **24**(2), 123–140, (1996).
- [3] L. Breiman, ‘Random forests’, *Machine Learning*, **45**(1), 5–32, (2001).
- [4] J. Demšar, ‘Statistical comparisons of classifiers over multiple data sets’, *Journal of Machine Learning Research*, **7**, 1–30, (2006).
- [5] T. G. Dietterich, ‘An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization’, *Machine Learning*, **40**(2), 139–158, (2000).
- [6] Y. Freund and R. E. Schapire, ‘A decision-theoretic generalization of on-line learning and an application to boosting’, *Journal of Computer and System Sciences*, **55**(1), 119–139, (August 1997).
- [7] T. K. Ho, ‘The random subspace method for constructing decision forests’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832–844, (1998).
- [8] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.
- [9] J. Li and H. Liu, ‘Ensembles of cascading trees’, in *3rd IEEE International Conference on Data Mining (ICDM’03)*, pp. 585–588, (2003).
- [10] C. Nadeau and Y. Bengio, ‘Inference for the generalization error’, *Machine Learning*, **52**(239–281), (2003).
- [11] R. E. Schapire, ‘The boosting approach to machine learning: An overview’, in *MSRI Workshop on Nonlinear Estimation and Classification*, (2002).
- [12] G. I. Webb, ‘Decision tree grafting from the all tests but one partition’, in *16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, Morgan Kaufmann, (1999).
- [13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2nd edn., 2005.

A Compression-Based Method for Stemmatic Analysis

Teemu Roos ¹ and Tuomas Heikkilä ² and Petri Myllymäki ¹

1 INTRODUCTION

Stemmatology studies relations among different variants of a text that has been gradually altered as a result of imperfectly copying the text over and over again. Underlying these variants there is what we could call a ‘family tree’, a graph representing the process of copying the text where each new version becomes a direct descendant of the exemplar(s) from which it is copied. The aim of stemmatic analysis is to reconstruct this family tree, known as the ‘stemma’, based on the surviving copies of the text. Applications are mainly in humanities, especially textual criticism, but the methods can be used to study the evolution of any symbolic objects, including chain letters [2] and computer viruses. We propose an algorithm for stemmatic analysis based on a minimum-information criterion and stochastic tree optimization. The intuitive idea behind compression-based approaches is that if a text can be significantly compressed, then the compression algorithm has found regularities which can be further exploited in an analysis such as ours. Our approach is related to phylogenetic reconstruction criteria such as maximum parsimony and maximum likelihood, and builds upon algorithmic techniques developed for bioinformatics. For a more detailed version see [14, 15].

2 A MINIMUM-INFORMATION CRITERION

In his seminal work on computer-assisted stemmatology, O’Hara used a parsimony method of the PAUP software [18] in Robinson’s Textual Criticism challenge [13]. For further applications of maximum parsimony and related method, see [8, 9, 17, 20] and the references therein.

Our compression-based *minimum information* criterion shares many properties of the maximum parsimony method. Both can also be seen as instances of the *minimum description length* (MDL) principle of Rissanen [12] which in turn is a formal version of Ockham’s razor. The minimum-information criterion measures the amount of information, or *code-length*, required to reproduce all the manuscripts by the process of copying and modifying the text under study. In order to describe a new version of an existing manuscript, one needs an amount of information that depends on both the amount and the type of modifications made. For instance, describing a deletion of a word or a change of word order requires less information than introducing a completely new expression.

In order to be concrete, we need a precise, numerical, and computable measure for the amount of information. The commonly accepted definition of the amount information in individual objects is Kolmogorov complexity, see [10], defined as the length of the shortest computer program to describe the given object. However, Kol-

mogorov complexity is defined only up to a constant, and fundamentally uncomputable. Therefore, in the spirit of a number of earlier authors [1, 2, 3, 4, 6, 11, 19], we approximate Kolmogorov complexity by using a compression program (`gzip`). In particular, given two strings, x and y , the amount of information in y conditional on x , denoted by $C(y | x)$ is given by the length of the compressed version of the concatenated string x, y minus the length of the compressed version of x alone. One of the advantages of using a string compression method that operates directly on the text is that only minimal preprocessing is required, contrary to most of the methods referred to above.

Let $G = (V, E)$ be an undirected graph where V is a set of nodes corresponding to the text variants, $E \subset V \times V$ is a set of edges. We require that the graph is a connected bifurcating tree, meaning that (i) each node has either one or three neighbors, and (ii) the tree is acyclic. Such a graph G can be made directed by picking any one of the nodes as the root and directing each edge away from the root. Given a directed graph \vec{G} , the total information cost of the tree is defined as

$$C(\vec{G}) = \sum_{v \in V} C(v | \text{Pa}(v)) \quad (1)$$

where the sum is over all the variants v , and $\text{Pa}(v)$ denotes the parent node of v unless v is the root in which case $\text{Pa}(v)$ is the empty string. For practical reasons (mainly for reduced computational cost) we make some modifications to this criterion, see the full version [15].

3 AN ALGORITHM FOR CONSTRUCTING STEMMATA

Since it is known that many of the text variants have been lost during the centuries between the time of the writing of the first versions and present time, it is not realistic to build a tree of only the available variants that we have as our data. The common way (in phylogeny) of handling this problem is to include in the tree a number of ‘hidden’ nodes, i.e., nodes representing individuals whose characteristics are unobserved. We construct bifurcating trees that have N observed nodes as leafs, and $N - 2$ hidden nodes as the interior nodes.

Evaluating the criterion (1) now involves the problem of dealing with the hidden nodes. Without knowing the values of v , it is not possible to compute $C(v | \text{Pa}_i(v))$. We solve this problem by searching simultaneously for the best tree structure \vec{G} and for the optimal contents of the hidden nodes with respect to criterion (1). Perhaps surprisingly, given a tree structure, finding the optimal contents is feasible. The method for efficiently optimizing the contents of the hidden nodes is an instance of dynamic programming (or ‘elimination’ in graphical models) and called ‘the Sankoff algorithm’ [5] or ‘Felsenstein’s algorithm’ [16].

There still remains the problem of finding the tree structure, which together with corresponding optimal contents of the hidden nodes

¹ Complex Systems Computation Group, Helsinki Institute for Information Technology, Finland, email: firstname.lastname@cs.helsinki.fi

² Dept. of History, University of Helsinki, Finland, email: tuomas.m.heikkila@helsinki.fi

minimizes criterion (1). The obvious solution, trying all possible tree structures and choosing the best one, fails because for N leaf nodes, the number of possible bifurcating trees is exponentially large (see [5]). Instead, we have to resort to heuristic search, trying to find as good a tree as possible in the time available. We use a simulated annealing algorithm that starts with an arbitrary tree and iteratively tries to improve it by small random modification, such as exchanging the places of two subtrees.

4 RESULTS AND CONCLUSIONS

We illustrate the behavior of the method by an artificial example in Fig. 1. Assume that we have observed five pieces of text, shown at the tips of the tree's branches. One of the trees — not the only one — minimizing the information cost with total cost of 44 units (bytes) is drawn in the figure. The sum of the (unconditional) complexities of the four words in the top-most string (“*sanctus henricus ex Anglia*”) is equal to $8 + 9 + 3 + 7 = 27$, which happens to coincide with the length of the string, including spaces and a finishing newline. The changes, labeled by numbers 1–5 in the figure, yield $5 + 3 + 3 + 3 = 17$ units of cost. Thus the total cost of the tree equals $27 + 17 = 44$ units.

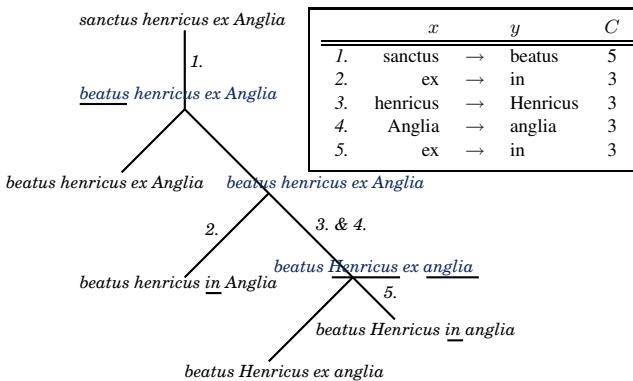


Figure 1. An example tree obtained with the compression-based method for the five strings at the tips of the branches. Changes are underlined and numbered. Costs of changes are listed in the box; no cost is incurred by a transition that leaves the string unchanged, i.e., $C(y | x) = 0$ if $y = x$. Best reconstructions at interior nodes are shown at the branching points.

In our main experiment, reported in the full version [15], we applied the presented method to the tradition of the legend of St. Henry of Finland³, of which some fifty manuscripts are known. Even for such a moderate number, manual stemma reconstruction is prohibitive due to the vast number of potential explanations, and the obtained stemma is the first attempt at a complete stemma of the legend of St. Henry. The relationships discovered by the method are largely supported by more traditional analysis in earlier work. Moreover, our results have pointed out groups of manuscripts not noticed in earlier manual analysis.

We are currently carrying out controlled experiments with artificial data with known ‘ground-truth’ solution to which the results can

³ St. Henry is a key figure of the Finnish Middle Ages. According to the medieval tradition, he was the Bishop of Uppsala (Sweden), and one of the leaders of a Swedish expedition to Finland around 1155, during which he was murdered. The oldest text concerning St. Henry is his legend written in Latin by the end of the 13th century at the very latest. For identification of the sources as well as a modern edition of the legend see [7].

be compared. Outside historical and biological applications, analysis of computer viruses is an interesting future research topic.

ACKNOWLEDGEMENTS

This work has significantly benefited from discussions with Tommi Mononen and Kimmo Valtonen at HIIT, and Prof. Paul Vitányi and Rudi Cilibrasi at CWI. This work was supported in part by IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

REFERENCES

- [1] D. Benedetto, E. Caglioti, and V. Loreto, ‘Language trees and zipping’, *Physical Review Letters*, **88**(4), 048702–1–048702–4, (2002).
- [2] C.H. Bennett, M. Li, and B. Ma, ‘Chain letters and evolutionary histories’, *Scientific American*, 76–81, (November 2003).
- [3] X. Chen, S. Kwong, and M. Li, ‘A compression algorithm for DNA sequences and its applications in genome comparison’, in *Genome Informatics*, eds., K. Asai, S. Miyano, and T. Takagi, Tokyo, (1999). Universal Academy Press.
- [4] R. Cilibrasi and P.M.B. Vitányi, ‘Clustering by compression’, *IEEE Transactions on Information Theory*, **51**(4), 1523–1545, (2005).
- [5] J. Felsenstein, *Inferring phylogenies*, Sinauer Associates, Sunderland, Massachusetts, 2004.
- [6] S. Grumbach and F. Tahi, ‘A new challenge for compression algorithms: genetic sequences’, *Journal of Information Processing and Management*, **30**(6), 875–866, (1994).
- [7] T. Heikkilä, *Pyhänen Henrikin legenda* (in Finnish), Suomalaisen Kirjallisuuden Seuran Toimituksia 1039, Helsinki, 2005.
- [8] C.J. Howe, A.C. Barbrook, M. Spencer, P. Robinson, B. Bordalejo, and L.R. Mooney, ‘Manuscript evolution’, *Trends in Genetics*, **17**(3), 147–152, (2001).
- [9] A.-C. Lantin, P. V. Baret, and C. Macé, ‘Phylogenetic analysis of Gregory of Nazianzus’ Homily 27’, in *7èmes Journées Internationales d’Analyse statistique des Données Textuelles*, eds., G. Purnelle, C. Fairon, and A. Dister, pp. 700–707, Louvain-la-Neuve, (2004).
- [10] M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd. Ed., Springer-Verlag, New York, 1997.
- [11] D. Loewenstein, H. Hirsh, P. Yianilos, and M. Noordewier, ‘DNA sequence classification using compression-based induction’, Technical Report 95–04, DIMACS, (1995).
- [12] J. Rissanen, ‘Modeling by shortest data description’, *Automatica*, **14**, 465–471, (1978).
- [13] P. Robinson and R.J. O’Hara, ‘Report on the textual criticism challenge 1991’, *Bryn Mawr Classical Review*, **3**(4), 331–337, (1992).
- [14] T. Roos, T. Heikkilä, R. Cilibrasi, and P. Myllymäki, ‘Compression-based stemmatology: A study of the legend of St. Henry of Finland’, Technical Report HIIT-2005-3, Helsinki Institute for Information Technology.
- [15] T. Roos, T. Heikkilä, and P. Myllymäki. A compression-based method for stemmatic analysis, 2006. Full version available at http://www.cs.helsinki.fi/teemu.roos/pub/ecai06_full.pdf.
- [16] A. Siepel and D. Haussler, ‘Phylogenetic estimation of context-dependent substitution rates by maximum likelihood’, *Molecular Biology and Evolution*, **21**(3), 468–488, (2004).
- [17] M. Spencer, K. Wachtel, and C.J. Howe, ‘The Greek Vorlage of the Syra Harclensia: A comparative study on method in exploring textual genealogy’, *TC: A Journal of Biblical Textual Criticism*, **7**, (2002).
- [18] D.L. Swofford. PAUP*: Phylogenetic analysis using parsimony (*and other methods). version 4., 2003.
- [19] J.-S. Varre, J.-P. Delahaye, and É. Rivals, ‘The transformation distance: a dissimilarity measure based on movements of segments’, in *Proceedings of German Conference on Bioinformatics*, Koel, Germany, (1998).
- [20] E. Wattel and M.P. van Mulken, ‘Weighted formal support of a pedigree’, in *Studies in Stemmatology*, eds., P. van Reenen and M.P. van Mulken, 135–169, Benjamins Publishing, Amsterdam, (1996).

Patch Learning for Incremental Classifier Design

Rudy Sicard¹ and Thierry Artières² and Eric Petit¹

Abstract. We present a learning algorithm for nominal data. It builds a classifier by adding iteratively a simple patch function that modifies the current classifier. Its main advantage lies in the possibility to learn every patch function parameters optimally from the Bayesian point of view hence avoiding overtraining.

1 INTRODUCTION

We present a learning algorithm for nominal data for designing complex classifiers while avoiding overtraining limitation, which is useful for small training sets and high dimensional data. Bayesian Model Averaging (BMA) provides a theoretical solution for avoiding overtraining [1] through computing expected probabilities by summing over all model parameters values. Unfortunately BMA is usually intractable so that approximations were proposed like Point Estimate Approximation (PEA) [2]. PEA consists in replacing the expectation over parameter value by a single term computed for one particular parameter value, Maximum A Posteriori (MAP) estimation is a typical example. However, MAP requires an ad-hoc (since it depends on the data and the task) parameter prior distribution that favours simple classifiers. Other approximation schemes have been proposed in the past such as Kullback Leibler (KL) projection [3] which has been used up to now for simple classifiers only. Building on KL projection we propose to learn incrementally a complex classifier in a way that is close to Boosting [4], by adding iteratively simple patch functions, where each patch function slightly modifies the current classifier. The major idea of our approach is to use simple enough patch functions so that above approximation scheme (and then optimal learning) is tractable. We focus on patch functions defined with a single parameter. Depending of the nature of the patch functions the built classifier resembles standard classifiers like Naïve Bayes (NB), logistic classification or other standard techniques. We first present what a patch function is and its optimal learning. Then we discuss incremental learning of complex classifiers. Lastly, we report experimental results that show the generalization ability and the accuracy of our technique on various benchmark datasets.

2 PATCH DESIGN AND LEARNING

For the sake of clarity we focus here on a simple problem with two classes but the extension to a more general multi-class problem is straightforward. In our mind a patch is an elementary function that modifies the behaviour of a current probabilistic classifier, F , that takes as input a vector x (of d nominal features) and outputs class posterior probabilities $\{P_F(C_j/x), j = 1, 2\}$. In the following a patch

is defined with one parameter α and a Boolean test T . It is used to modify F into $F'_{T,\alpha}$ by changing the behaviour of F for inputs that verify T only. New classifier $F'_{T,\alpha}$ is defined as follows:

$$\begin{aligned} \text{if } T(x) \text{ is true} & \begin{cases} P_{F'_{T,\alpha}}(C_1/x) = 1/z(x) \times \alpha \times P_F(C_1/x) \\ P_{F'_{T,\alpha}}(C_2/x) = 1/z(x) \times (1-\alpha) \times P_F(C_2/x) \end{cases} \\ \text{else} & P_{F'_{T,\alpha}}(C_j/x) = P_F(C_j/x), j = 1, 2 \end{aligned} \quad (1)$$

where $z(x) = (\alpha \times P_F(C_1/x) + (1-\alpha) \times P_F(C_2/x))$ is a normalization factor. Examples of test functions would be: $T(x)$ is true if the i^{th} feature of x has a particular value v (single feature patch) or $T(x)$ is true if the i^{th} feature has a value u and the j^{th} feature has value v (double feature patch). Learning a patch is done through Bayesian learning. Using posterior class probability estimated with $F'_{T,\alpha}$, the posterior probability for parameter α (thus indexed by F'_T) is:

$$P_{F'_T}(\alpha / D) \propto Z(D) \times P(\alpha) \prod_{i/T(x_i)=\text{true}} P_{F'_{T,\alpha}}(y_i / x_i) \quad (2)$$

where $Z(D)$ is a normalization factor, $P(\alpha)$ is a prior distribution and the product ranges over training samples x_i (whose class is y_i) that verify the test T . The optimal Bayesian classifier may be defined through Bayesian Model Averaging [1]:

$$P_{F'_T}(C_i / D, x) = \int_{\alpha=0}^1 p_{F'_{T,\alpha}}(C_i / x) p_{F'_T}(\alpha / D) d\alpha \quad (3)$$

Given x , this BMA distribution minimizes the distance (measured as the KL divergence [3]) with the true distribution by averaging over α [1]. Computing this distribution may be done analytically for well chosen priors but this is expensive. Here, we seek to approximate this term with a PEA strategy which aims at replacing summation like in Eq. (4) with one term $p_{F'_{T,\alpha}}(C_i / x)$. Following

[3] we chose to seek a PEA approximation that is as close as possible to this optimal BMA classifier, w.r.t. KL divergence. Since the optimal PEA may depend on x , i.e. may be obtained with possibly different α , the solution is to minimize the expected KL divergence, over x , of PEA with respect to BMA [2]. We do not detail here methods to perform this minimization for space reasons but this is not a critical point, it is performed numerically (note that the criterion is convex and hence does no exhibit local minima).

3 COMPLEX CLASSIFIER CONSTRUCTION

Using the patch learning routine discussed above, one can build incrementally complex classifiers by adding sequentially a number of patches, starting from e.g. an initial neutral classifier ($\forall x, \forall j = 1..2, P(C_j/x) = 0.5$). Let $\{(T_i, \alpha_i)\}_{i=1..N}$ be the successive

¹ France Télécom – R&D/ TECH/IDEA/TIPS – France.

² LIP6 - Université Pierre et Marie Curie, France.

patches that are learned for building a classifier F , each patch is defined by a Boolean test T_i and a parameter value α_i . Then, the final classifier may be written in a standard form:

$$P(C_1 | x) = Z(x) \times \exp(W^t U(x)) \quad (4)$$

where $W = (\log(\alpha_i))_{i=1..N}$ is the parameter vector for classifier F and $U(x) = (T_i(x))_{i=1..N}$ is the feature vector corresponding to input x (with $T_i(x) = 0/1$). Depending on the nature of the patches, the final classifier may correspond to various classical models. For instance, when considering single feature patches, decision boundaries are linear and have similar form as for NB or logistic classification models. Training consists in applying iteratively a number of patches (eventually all possible patches). In the case of single feature patches, learning algorithm resumes to:

For each feature a and for each feature value v_a
Add the patch with test $T(x) = [x.a == v_a]$ and
optimal α parameter as discussed in section 2.

As usual, one may design more complex classifier through the use of complex features and the addition of many features but this may raise overtraining problem. At the opposite, using our optimal approach described in section 2 should overcome this problem. The counterpart of this generalization ability is that our approach cannot be shown to be the globally optimal. It is locally optimal only in that each patch is learned optimally. For instance, the final classifier will depend on the order the patches are added. However, we believe this global non optimality must be considered together with its generalization power.

4 EXPERIMENTS

We performed experiments on a number of UCI benchmark datasets² and compared our approach with standard techniques using the Weka package [5]. We report here two series of experiments. First we investigated overtraining robustness and compared our patch learning algorithm with single and double features patches to NB and Logistic classification (Lg). Figure 1 shows their performances averaged over 100 runs as a function of the training set size with a logarithmic scale to focus on small sizes on two datasets whose settings are detailed in Table 1. It may be seen first that NB is significantly less accurate than other approaches, second that our approach (with single or double patches) significantly outperforms others for a wide range of training set sizes, third that as more training data becomes PL2 and Lg tend to behave similarly while slightly outperforming PL1 on the kr-kp dataset. This was clearly expected. With small training set size our approach naturally outperforms standard methods such as Lg or NB. As more training data becomes available all models (except NB) perform similarly on vote dataset since it is almost linearly separable while PL1 perform slightly lower than richer (PL2) and globally optimally learned (Lg) models on the more difficult kr-kp dataset. Note that Lg may slightly outperform patch learning in some cases as we will see, which may come from the fact that Lg is globally optimal. The second series of experiments compare patch learning to standard methods, NB, Lg and AODE [6] on a number of UCI benchmark datasets (Table 1) with a standard training set size and a 10-fold cross validation procedure. It may be seen that, except for the zoo dataset, double feature

patches outperform single feature patches, that single patches is comparable or better than standard techniques and that double features patches outperforms all other methods in average. Hence patch learning is not only able to learn complex classifier while avoiding overtraining (as in Fig. 1), it also allows learning efficient classifiers that compare well to more standard techniques.

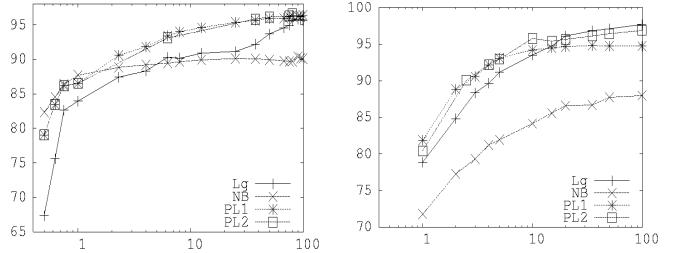


Figure 1. Comparison of generalization ability for Naive Bayes (NB), Logistic Classification (Lg), Patch learning with single (PL1) and double (PL2) feature patches. Performances are plotted as a function of the size of the training set (percentage of whole dataset size) on the UCI *vote* dataset (left) and *kr-kp* dataset (right) whose settings are detailed in Table 1.

Table 1. Comparison of error-rates for Naïve Bayes (NB), AODE, Logistic classification (Lg) and patch learning (PL1 and PL2) on various datasets from UCI⁴. The last row indicates mean error-rate relatively to NB performance. Note that the number of classes #C, the dimension of data d and the training set size TSS (#samples) are indicated for each dataset.

Data set	#C	d	TSS	NB	AODE	PL1	PL2	Lg
Audiology	23	70	226	26.6	26.6	19.1	16.4	17.3
Hepatitis	2	18	155	17.5	16.8	18.1	16.8	22.7
Kr-kp	2	37	3196	12.1	8.8	5.4	2.9	2.5
Mushroom	2	22	8124	4.2	0.05	0.05	0.0	0.0
P-tumor	23	18	329	49.9	50.4	53.4	54.3	55.1
Splice	3	60	3190	4.7	3.9	3.6	3.6	9.0
Vote	2	16	435	9.9	5.7	3.9	3.2	3.9
Zoo	7	17	101	6.8	4.9	2.8	3.8	5.8
Average performance				1.00	0.73	0.61	0.57	0.64

5 CONCLUSION

We presented an incremental learning algorithm for nominal data that allows designing iteratively complex classifiers. It relies on the optimal learning of simple patch functions. Although the resulting classifier is not globally optimal it exhibits good generalization behaviour with small training sets while comparing well to standard techniques with reasonable training set size. Note that up to now we used patches involving single and double features only since using more complex patches is computationally more expensive. This is still under investigation.

6 REFERENCES

- [1] Hoeting J., Madigan D., Raftery A. and Volinsky C., *Bayesian Model Averaging*, Statistical Science 14, 382-401, 1999.
- [2] Herbrich R., Graepel T., Campbell C., *Bayes Point Machines*, Journal of Machine Learning Research, 1:245-279, 2001.
- [3] Snelson, E. and Ghahramani, Z., *Compact approximations to Bayesian predictive distributions*, ICML 2005.
- [4] Tresp W., Committee machines, in Handbook for Neural Network Signal Processing, Y.H. Hu and J-N. Hwang eds. CRC Press, 2001.
- [5] Witten I.H. and Eibe F., Data Mining: Practical machine learning tools and techniques, Morgan Kaufman, 2005.
- [6] Webb G., Boughton J., Wang Z., *Not So Naive Bayes: Aggregating One-Dependence Estimators*, Machine Learning 58(1): 5-24, 2005.

² Newman D.J. and al., UCI repository of machine learning databases, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, University of California.

Version Space Support Vector Machines

E.N. Smirnov¹, I.G. Sprinkhuizen-Kuyper¹, G.I. Nalbantov² and S. Vanderlooy¹

1 Introduction

The task of reliable classification is to determine if a particular instance classification is reliable. There exist two approaches to the task: the Bayesian framework [3] and the typicalness framework [5]. Although both frameworks are useful, the Bayesian framework can be misleading and the typicalness framework is classifier dependent.

To overcome these problems we argue to use version spaces (VSs) [3] for reliable classification. The key idea is to construct VSs containing hypotheses of the target class or of its close approximations. In this way the unanimous-voting classification rule of VSs does not misclassify instances; i.e., instance classifications become reliable.

To construct VSs with the property above we propose an approach extending the volumes of VSs s.t. instance misclassifications are blocked. The approach and VSs are realized using support vector machines (SVMs) [4]. The combination is called version space support vector machines (VSSVMs). We show that VSSVMs are able to outperform the existing approaches to reliable classification.

2 Task of Reliable Classification

Consider training instances $\mathbf{x}_i \in \mathbb{R}^n$ with labels $y_i \in \{-1, +1\}$ of a target class s.t. \mathbf{x}_i is in set I^+ (I^-) if $y_i = +1$ ($y_i = -1$). Given a space H of hypotheses h ($h : \mathbb{R}^n \rightarrow \{-1, +1\}$), the task is to find $h \in H$ classifying correctly instances in \mathbb{R}^n . If correct classification of $\mathbf{x} \in \mathbb{R}^n$ is not possible, \mathbf{x} is unclassified (indicated by label 0).

3 Version Spaces

Given a hypothesis space H and training data $\langle I^+, I^- \rangle$, version space $VS(I^+, I^-)$ is the set of hypotheses consistent with $\langle I^+, I^- \rangle$:

$$VS(I^+, I^-) = \{h \in H | cons(h, \langle I^+, I^- \rangle)\},$$

where $cons(h, \langle I^+, I^- \rangle) \leftrightarrow (\forall \mathbf{x}_i \in I^+ \cup I^-) y_i = h(\mathbf{x}_i)$.

The unanimous-voting rule of VSs classifies instance \mathbf{x} as follows:

$$y = \begin{cases} +1 & VS(I^+, I^-) \neq \emptyset \wedge (\forall h \in VS(I^+, I^-)) h(\mathbf{x}) = +1, \\ -1 & VS(I^+, I^-) \neq \emptyset \wedge (\forall h \in VS(I^+, I^-)) h(\mathbf{x}) = -1, \\ 0 & \text{otherwise. (instance } \mathbf{x} \text{ is left unclassified)} \end{cases}$$

The volume $V(VS(I^+, I^-))$ of $VS(I^+, I^-)$ is the set of instances not classified by the unanimous-voting rule ($y = 0$). By theorem 1 the rule can be implemented by testing VSs for collapse [2].

Theorem 1 If $VS(I^+, I^-) \neq \emptyset$, then for each instance \mathbf{x} :

$$\begin{aligned} (\forall h \in VS(I^+, I^-)) h(\mathbf{x}) = +1 &\leftrightarrow VS(I^+, I^- \cup \{\mathbf{x}\}) = \emptyset, \\ (\forall h \in VS(I^+, I^-)) h(\mathbf{x}) = -1 &\leftrightarrow VS(I^+ \cup \{\mathbf{x}\}, I^-) = \emptyset. \end{aligned}$$

¹ MICC-IKAT, Universiteit Maastricht, Maastricht 6200 MD, The Netherlands, email: {smirnov, kuyper, s.vanderlooy}@cs.unimaas.nl

² ERIM, Erasmus University Rotterdam, Rotterdam 3000 DR, The Netherlands, email: nalbantov@few.eur.nl

Testing VSs for collapse is the consistency problem [2]. The problem is to check the existence of a hypothesis $h \in H$ consistent with data. Hence, the VSs unanimous-voting rule can be implemented by any algorithm for the consistency problem (consistency algorithm).

4 Volume-Extension Approach

VSs are sensitive to the class noise in the training data and the expressiveness of hypothesis space H [3]. The expressiveness of H indicates if the hypothesis h_t of the target class is in H . We analyze VS instance classification with these two factors.

Case 1: no class noise and expressive hypothesis space. In this case $h_t \in VS(I^+, I^-)$. Thus, if instance \mathbf{x} is classified by $VS(I^+, I^-)$, \mathbf{x} is classified by h_t ; i.e., \mathbf{x} is classified correctly (reliably).

Case 2: class noise. In this case the set I^+ (I^-) is a union of a noise-free set I_f^+ (I_f^-) and a noisy set I_n^+ (I_n^-). The noisy data $\langle I_n^+, I_n^- \rangle$ cause removal of version space $NVS = \{h \in VS(I_f^+, I_f^-) | \neg cons(h, \langle I_n^+, I_n^- \rangle)\}$ from $VS(I_f^+, I_f^-)$. Thus, $VS(I^+, I^-)$ classifies correctly (reliably) instances classified by $VS(I_f^+, I_f^-)$, but errs on some instances in the volume of NVS .

Case 3: inexpressive hypothesis space. Since $h_t \notin H$, some instances \mathbf{x} can be misclassified by all hypotheses in $VS(I^+, I^-)$.

The volume-extension approach is a new approach for cases 2-3. If $VS(I^+, I^-) \subseteq H$ misclassifies, then the approach is to find a new hypothesis space H' s.t. the volume of $VS'(I^+, I^-) \subseteq H'$ grows and blocks misclassifications. It is applied using theorem 2.

Theorem 2 Consider hypothesis spaces H and H' s.t. for all $\langle I^+, I^- \rangle$ if there is $h \in H$ consistent with $\langle I^+, I^- \rangle$, then there is $h' \in H'$ consistent with $\langle I^+, I^- \rangle$. Then, for all $\langle I^+, I^- \rangle$ the volume $V(VS(I^+, I^-))$ is a subset of the volume $V(VS'(I^+, I^-))$.

Below we apply the volume-extension approach for cases 2-3.

Case 2: since the volume V of NVS is the error region, we search for H' s.t. the volume of $VS'(I^+, I^-)$ comprises maximally V ;

Case 3: since the cause of misclassifications is that $h_t \notin H$, we search for H' s.t. $VS'(I^+, I^-)$ includes more hypotheses approximating h_t . This means that if \mathbf{x} is misclassified, we define H' s.t. $VS'(I^+, I^-)$ includes a hypothesis classifying \mathbf{x} as h_t does. Thus, \mathbf{x} is not classified, so the misclassification is blocked.

We conclude that our approach can block misclassifications for cases 2-3. This means that VSs can be used for reliable classification.

5 Support Vector Machines

Support Vector Machines (SVMs) construct hyperplanes $h(p, C, \langle I^+, I^- \rangle)$ in the hypothesis space $H(p)$ of oriented hyperplanes s.t. the margin between training sets I^+ and I^- is maximized given kernel parameter p and parameter C that controls the trade-off between the margin and the training errors [4].

6 Version Space Support Vector Machines

The key idea of VSSVMs is to use a SVM as a consistency algorithm. By theorem 1 we need a consistency test only for $\langle I^+ \cup \{x\}, I^- \rangle$ and $\langle I^+, I^- \cup \{x\} \rangle$. Since a SVM is not a consistency algorithm in $H(p)$ [4], we define a hypothesis space $H(p, C, \langle I^+, I^- \rangle)$ for which SVM is a consistency algorithm for $\langle I^+ \cup \{x\}, I^- \rangle$ and $\langle I^+, I^- \cup \{x\} \rangle$.

Definition 1 Given parameters p and C and data $\langle I^+, I^- \rangle$, if $\text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+, I^- \rangle)$, then $H(p, C, \langle I^+, I^- \rangle)$ equals:

$$\begin{aligned} & \{h \in H(p) | h = h(p, C, \langle I^+, I^- \rangle) \vee \\ & (\exists x)(h = h(p, C, \langle I^+ \cup \{x\}, I^- \rangle) \wedge \text{cons}(h, \langle I^+ \cup \{x\}, I^- \rangle)) \vee \\ & (\exists x)(h = h(p, C, \langle I^+, I^- \cup \{x\} \rangle) \wedge \text{cons}(h, \langle I^+, I^- \cup \{x\} \rangle)), \\ & \text{otherwise, } H(p, C, \langle I^+, I^- \rangle) = \emptyset. \end{aligned}$$

SVM is a consistency algorithm if the property below holds.

Definition 2 SVM has the instance-consistency property w.r.t. data $\langle I^+, I^- \rangle$ if and only if for any instance x :

- (i) if $h(p, C, \langle I^+ \cup \{x\}, I^- \rangle)$ is inconsistent with $\langle I^+ \cup \{x\}, I^- \rangle$, then for all x' $h(p, C, \langle I^+ \cup \{x'\}, I^- \rangle)$ and $h(p, C, \langle I^+, I^- \cup \{x'\} \rangle)$ are inconsistent with $\langle I^+ \cup \{x\}, I^- \rangle$;
- (ii) if $h(p, C, \langle I^+, I^- \cup \{x\} \rangle)$ is inconsistent with $\langle I^+, I^- \cup \{x\} \rangle$, then for all x' $h(p, C, \langle I^+ \cup \{x'\}, I^- \rangle)$ and $h(p, C, \langle I^+, I^- \cup \{x'\} \rangle)$ are inconsistent with $\langle I^+, I^- \cup \{x\} \rangle$.

Theorem 3 sets the SVM consistency test in $H(p, C, \langle I^+, I^- \rangle)$.

Theorem 3 If the instance-consistency property holds and $H(p, C, \langle I^+, I^- \rangle) \neq \emptyset$, then for each instance x we have:

$$\begin{aligned} & (\exists h \in H(p, C, \langle I^+, I^- \rangle)) \text{cons}(h, \langle I^+ \cup \{x\}, I^- \rangle) \leftrightarrow \\ & [\text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle) \vee \\ & \text{cons}(h(p, C, \langle I^+ \cup \{x\}, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle)], \\ & (\exists h \in H(p, C, \langle I^+, I^- \rangle)) \text{cons}(h, \langle I^+, I^- \cup \{x\} \rangle) \leftrightarrow \\ & [\text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+, I^- \cup \{x\} \rangle) \vee \\ & \text{cons}(h(p, C, \langle I^+, I^- \cup \{x\} \rangle), \langle I^+, I^- \cup \{x\} \rangle)]. \end{aligned}$$

By theorem 3 to test if there is a hyperplane in $H(p, C, \langle I^+, I^- \rangle)$ consistent with $\langle I^+ \cup \{x\}, I^- \rangle$ test if either of the SVM hyperplanes $h(p, C, \langle I^+, I^- \rangle)$ and $h(p, C, \langle I^+ \cup \{x\}, I^- \rangle)$ is consistent with $\langle I^+ \cup \{x\}, I^- \rangle$. Testing if there is a hyperplane in $H(p, C, \langle I^+, I^- \rangle)$ consistent with $\langle I^+, I^- \cup \{x\} \rangle$ is analogous.

Given $H(p, C, \langle I^+, I^- \rangle)$, VSSVMs are defined as follows:

Definition 3 Given data $\langle I^+, I^- \rangle$ s.t. $I^+ \supseteq I^+$ and $I^- \supseteq I^-$, the version space support vector machine $VS_C^p(I^+, I^-)$ is:

$$VS_C^p(I^+, I^-) = \{h \in H(p, C, \langle I^+, I^- \rangle) | \text{cons}(h, \langle I^+, I^- \rangle)\}.$$

The classification algorithm of VSSVMs realizes the unanimous-voting rule (figure 1). It starts classifying instance x by first building the SVM hyperplane $h(p, C, \langle I^+, I^- \rangle)$. If $h(p, C, \langle I^+, I^- \rangle) = \emptyset$. Thus, $VS_C^p(I^+, I^-) = \emptyset$ and by the unanimous-voting rule 0 is returned. If $h(p, C, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+, I^- \rangle$, $VS_C^p(I^+, I^-) \neq \emptyset$. Hence, the algorithm tests if $h(p, C, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+ \cup \{x\}, I^- \rangle$. If so, $VS_C^p(I^+ \cup \{x\}, I^-) \neq \emptyset$ and the algorithm builds $h(p, C, \langle I^+, I^- \cup \{x\} \rangle)$. If $h(p, C, \langle I^+, I^- \cup \{x\} \rangle)$ is inconsistent with $\langle I^+, I^- \cup \{x\} \rangle$, by theorem 3 $VS_C^p(I^+, I^- \cup \{x\}) = \emptyset$. Since $VS_C^p(I^+ \cup \{x\}, I^-) \neq \emptyset$ and $VS_C^p(I^+, I^- \cup \{x\}) = \emptyset$, by theorem 1 class +1 is assigned to x . If class +1 is not assigned, we check analogously if class -1 can be assigned. If no class is assigned, 0 is returned.

```

Build a hyperplane  $h(p, C, \langle I^+, I^- \rangle)$ ;
if  $\neg \text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+, I^- \rangle)$  then return 0;
if  $\text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle)$  then
    Build hyperplane  $h(p, C, \langle I^+, I^- \cup \{x\} \rangle)$ ;
    if  $\neg \text{cons}(h(p, C, \langle I^+ \cup \{x\}, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle)$  then return +1;
if  $\text{cons}(h(p, C, \langle I^+, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle)$  then
    Build hyperplane  $h(p, C, \langle I^+ \cup \{x\}, I^- \rangle)$ ;
    if  $\neg \text{cons}(h(p, C, \langle I^+ \cup \{x\}, I^- \rangle), \langle I^+ \cup \{x\}, I^- \rangle)$  then return -1;
return 0.

```

Figure 1. The Classification Algorithm of VSSVMs.

The volume-extension approach for VSSVMs is applied for cases 2-3 using the parameter C of SVMs. Note that the probability that the SVM hyperplane $h(p, C, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+, I^- \rangle$ increases with C . Thus, for values C_1 and C_2 of C s.t. $C_1 < C_2$ the probability that $h(p, C_2, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+, I^- \rangle$ is higher than that of $h(p, C_1, \langle I^+, I^- \rangle)$. This implies by theorem 2 that the volume of VSSVMs increases with C .

Applying the approach means to find C s.t. $VS_C^p(I^+, I^-) \subseteq H(p, C, \langle I^+, I^- \rangle)$ classifies instances reliably. Since the volume of VSSVMs increases with C , we can find the minimal value for C in a validation process using binary search s.t. instances are classified reliably (correctly) and the volume of $VS_C^p(I^+, I^-)$ is minimized.

7 Experiments and Comparison

We tested VSSVMs using a polynomial kernel (VSSVM-P). We applied the volume-extension approach by incrementing the parameter C to find the maximal accuracy rate A_m and the maximal coverage rate C_m for A_m (coverage rate is the proportion of classified instances). Table 1 compares VSSVM-P with the naive Bayes classifier (NB) (Bayesian framework) and typicalness-based naive Bayes classifier (TNB) (typicalness framework). The comparison is made using A_m and C_m measured by the leave-one-out method. It shows that VSSVM-P outperform NB and TNB w.r.t. reliable classification.

Data Set	VSSVM-P		NB		TNB	
	C_m	A_m	C_m	A_m	C_m	A_m
labor	0.456	1.0	0.396	1.0	0.351	1.0
hepatitis	0.684	1.0	0.013	1.0	0.013	1.0
breast	0.158	1.0	0.072	1.0	0.079	0.921
sonar	0.221	1.0	0.053	1.0	0.005	1.0
colic	0.079	1.0	0.006	1.0	0.360	0.906
wisc.breast	0.687	1.0	0.017	1.0	0.005	1.0
ionosphere	0.706	1.0	0.168	1.0	0.120	1.0

Table 1. The maximal accuracy A_m and coverage C_m of VSSVM-P, NB, and TNB for 7 datasets from [1].

REFERENCES

- [1] C. Blake and C. Merz. UCI repository of ML databases, 1998.
- [2] H. Hirsh, N. Mishra, and L. Pitt, ‘Version spaces and the consistency problem’, *Artificial Intelligence*, **156**(2), 115–138, (2004).
- [3] T.M. Mitchell, *Machine learning*, McGraw-Hill, New York, NY, 1997.
- [4] V. Vapnik, *Statistical Learning Theory*, John Wiley, NY, 1998.
- [5] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*, Springer, New York, NY, 2005.

Meta-Typicalness Approach to Reliable Classification

E.N. Smirnov¹, S. Vanderlooy¹ and I.G. Sprinkhuizen-Kuyper¹

Abstract. We propose a meta-typicalness approach to apply the typicalness framework for any type of classifiers. The approach can be used to construct classifiers with specified classification performance. Experiments show that the approach results in classifiers that can outperform an existing typicalness-based classifier.

1 Introduction

Machine-learning classifiers were applied to many classification tasks [2, 4]. Nevertheless, only few classifiers were used in critical-domain applications [7]. This is due to the difficulty to determine if a classification assigned to a particular instance is reliable.

To decide if an instance classification is reliable we have two tasks:

- (T1) to obtain a confidence value for the classification; and
- (T2) to obtain a threshold on confidence values.

If the confidence value for a classification of an instance is greater than the threshold, the classification is considered as reliable. Otherwise, it is unreliable and the instance is left unclassified.

The typicalness framework was proposed for task T1 [5, 6, 8]. It provides confidence values for instance classifications under the assumption that the data are independent and identically distributed (iid). To apply the framework for a classifier an instance-strangeness function has to be constructed specific for that classifier. Since this is difficult, the framework applicability is restricted: it was applied so far only for SVMs [6] and the nearest-neighbor classifier [5].

To apply the typicalness framework for any type of classifiers we propose a meta-typicalness approach inspired by [1]. The approach is to learn a meta typicalness-based classifier predicting the correctness of each instance classification of a base classifier. Thus, the confidence values of the meta predictions are viewed as the confidence values of the instance classifications of the base classifier.

To obtain a threshold on the confidence values (task T2) for the meta-typicalness approach we propose a procedure that allows constructing classifiers with certain classification performance.

This paper is organized as follows. Section 2 introduces the classification task. The typicalness framework is given in section 3. Section 4 introduces the meta-typicalness approach. The procedure for classifier construction is presented in section 5. Section 6 describes experiments, and section 7 concludes the paper.

2 Classification Task

Let \mathbf{X} be an instance space and Y a class set. Training data D is a set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ of n labelled instances (\mathbf{x}_i, y_i) where instance \mathbf{x}_i is in \mathbf{X} and class y_i is in Y . Given a space H of hypotheses h ($h : \mathbf{X} \rightarrow Y$), the classification task is to find $h \in H$ that correctly classifies future, unseen instances.

¹ MICC-IKAT, Universiteit Maastricht, Maastricht 6200 MD, The Netherlands, email: {smirnov, kuyper, s.vanderlooy}@cs.unimaas.nl

3 The Typicalness Framework

The typicalness framework [5, 6, 8] assumes that training data D and an instance \mathbf{x}_{n+1} to be classified are iid drawn from the same unknown distribution. Given a classifier C , we compute the p -value for each class $y_{n+1} \in Y$ that C can assign to \mathbf{x}_{n+1} as follows:

$$p(y_{n+1}) = \frac{\#\{i : \alpha_i \geq \alpha_{n+1}\}}{n+1}$$

where α_i is the strangeness of instance (\mathbf{x}_i, y_i) in the bag $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{i-1}, y_{i-1}), (\mathbf{x}_{i+1}, y_{i+1}), \dots, (\mathbf{x}_{n+1}, y_{n+1})\}$. The class y_{n+1} with the largest p -value is the classification of \mathbf{x}_{n+1} . The typicalness (credibility) of this classification is the largest p -value and the confidence is one minus the second largest p -value.

To compute strangeness values α_i we need to construct an instance-strangeness function α for the classifier C used. Since this is difficult, the applicability of the typicalness framework is restricted.

4 Meta-Typicalness Approach

To apply the typicalness framework to any classifier we propose a meta-typicalness approach (see task T1). Consider a base classifier B trained on training data D . We learn a meta classifier M that predicts the reliability of each classification of B . Hence, the meta class set Y' of metadata D' for M consists of two meta classes: “0” (unreliable classification) and “1” (reliable classification). Metadata D' is formed using k -fold cross validation as follows: a labelled meta instance $(\mathbf{x}_i, y'_i) \in D'$ is formed from the labelled instance $(\mathbf{x}_i, y_i) \in D$ s.t. y'_i is equal to “1” if the classification of \mathbf{x}_i provided by classifier B is equal to y_i ; otherwise, y'_i equals “0”.

The classifiers B and M form one combined classifier denoted by $B:M$. The classification rule of $B:M$ is to assign a class y predicted by B to an instance \mathbf{x} if M decides that this classification is reliable (i.e., M assigns meta class “1” to \mathbf{x}); otherwise, \mathbf{x} is unclassified.

Assume that the classifier M is based on the typicalness framework and the classifier B is not (i.e., M outputs confidence values for classifications while B does not output such values). Then, we can use the p -values of M as the p -values for $B:M$ as follows. If B assigns a class y to an instance \mathbf{x} and M assigns meta class “1” to \mathbf{x} , then the meta-typicalness approach assigns the p -value p_1 of meta class “1” to the p -value of class y of B and the p -value p_0 of meta class “0” to p -values of all the remaining classes $Y \setminus \{y\}$. Thus, we can estimate the typicalness and confidence of each classification y of $B:M$. We note that p -values of the classes $Y \setminus \{y\}$ are overestimated if $|Y \setminus \{y\}| > 1$. Thus, the confidence estimate can be pessimistic.

5 Combined-Classifier Construction

To decide if a classification of a combined classifier $B:M$ is reliable, we threshold the classification-confidence values of $B:M$ (see task

T2). This is equivalent to thresholding the confidence values of M . By theorem 1 to build $B:M$ with accuracy \mathcal{A} we threshold the confidence values of M s.t. the precision of M becomes equal to \mathcal{A} ².

Theorem 1 [3] *The accuracy \mathcal{A} of the combined classifier $B:M$ equals the precision \mathcal{PR} of the meta classifier M .*

To build a meta classifier M with precision \mathcal{PR} we propose a procedure using ROC convex hull (ROCCH) [7] (figure 1). If meta class “1”(“0”) is the positive(negative) class, the procedure is as follows:

- (1) build the ROCCH of M using the confidence ratio $\frac{(1-p_0)}{(1-p_1)}$;
- (2) construct the iso-precision line of the desired precision \mathcal{PR} given by $tpr = \frac{\mathcal{PR}}{1-\mathcal{PR}} \frac{N}{P} fpr$, where $tpr(fpr)$ is the true(false) positive rate, and $P(N)$ is the number of positive(negative) meta instances;
- (3) determine the value R_I of the ratio $\frac{(1-p_0)}{(1-p_1)}$ in the intersection point I of the ROCCH and the iso-precision line.

If the confidence ratio $\frac{(1-p_0)}{(1-p_1)}$ of the meta classifications are thresholded by the value R_I , then the precision of M is \mathcal{PR} [7]. Thus, by theorem 1 the accuracy \mathcal{A} of $B:M$ will be equal to \mathcal{PR} .

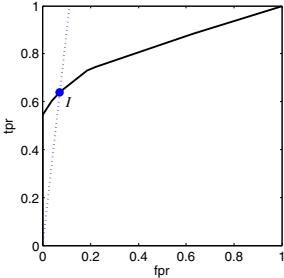


Figure 1. ROCCH of the typicalness-based nearest-neighbor classifier [5] trained on meta data obtained by a naive Bayes classifier on the Wisconsin breast-cancer data [2]. ROCCH is intersected by iso line of 90% precision.

By theorem 1 the accuracy \mathcal{A} of the combined classifier $B:M$ is maximized if the precision of M is maximized. This precision is maximized for the iso-precision line with slope equal to the slope of the line segment of the ROCCH starting from the origin. The highest point of this segment maximizes the number of covered meta instances (see point $(0, 0.56)$ of ROCCH in figure 1). Thus, in this point by theorem 1 the accuracy and the coverage of $B:M$ are maximized. They are denoted by \mathcal{A}_m and \mathcal{C}_m , respectively.

6 Experiments

We tested the meta-typicalness approach using the combined classifier $NB:TCMNN$ where NB is the naive-Bayes classifier [4] and $TCMNN$ is a typicalness-based nearest-neighbor classifier [5]. Note that NB does not compute classification-confidence values in contrast with $TCMNN$. The classification-confidence values of $NB:TCMNN$ were formed according to the meta-typicalness approach.

The metadata for $TCMNN$ in $NB:TCMNN$ were generated in an internal leave-one-out process. By thresholding classification-confidence values of $NB:TCMNN$ we constructed coverage/accuracy graphs of $NB:TCMNN$ using 10-fold cross validation. A comparison of the graphs with those of $TCMNN$ shows that $NB:TCMNN$ outperforms $TCMNN$ on the majority of the datasets. Since the space is not

² Classifiers M and $B:M$ may not classify all the instances. The accuracy and precision of these classifiers are defined only for the instances they are able to classify. Thus, coverage defined as the proportion of the classified instances is an important statistic for these classifiers.

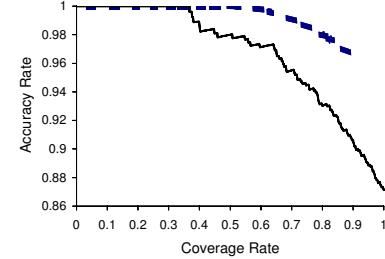


Figure 2. The Coverage/Accuracy graphs of $NB:TCMNN$ (---) and $TCMNN$ (—) for the Wisconsin breast-cancer data.

enough (we show only the graphs for one dataset in figure 2), we compare both classifiers in terms of maximal accuracy \mathcal{A}_m and coverage \mathcal{C}_m in table 1. A comparison of these statistics for 12 datasets [2] shows that $NB:TCMNN$ outperforms $TCMNN$ for 7 datasets.

Data Set	$NB:TCMNN$		$TCMNN$		NB \mathcal{A}_{NB}
	\mathcal{C}_m	\mathcal{A}_m	\mathcal{C}_m	\mathcal{A}_m	
audiology	0.08	1.0	0.01	1.0	0.73
wisc. breast cancer	0.51	1.0	0.35	1.0	0.96
colic	0.15	1.0	0.04	1.0	0.78
diabetes	0.01	1.0	0.02	1.0	0.76
heart-statlog	0.10	1.0	0.14	1.0	0.84
hepatitis	0.25	1.0	0.34	1.0	0.85
ionosphere	0.22	1.0	0.44	1.0	0.83
iris	0.63	1.0	0.86	1.0	0.96
lymphography	0.12	1.0	0.01	1.0	0.83
soybean	0.08	1.0	0.02	1.0	0.93
vote	0.14	1.0	0.11	1.0	0.90
zoo	0.22	1.0	0.03	1.0	0.95
<i>average</i>	0.27	1.0	0.20	1.0	0.86

Table 1. Maximal accuracy \mathcal{A}_m and coverage \mathcal{C}_m of the classifiers $NB:TCMNN$ and $TCMNN$. \mathcal{A}_{NB} is the accuracy of the NB classifier.

7 Conclusion

In this paper we proposed a meta-typicalness approach that allows the typicalness framework to be applied to any classifier. We showed how the approach can be used for constructing classifiers with certain classification performance. The experiments showed that the approach allows constructing combined classifiers that are able to outperform the $TCMNN$ classifier.

REFERENCES

- [1] S. Bay and M. Pazzani, ‘Characterizing model errors and differences’, in *Proceedings of the 17th International Conference on Machine Learning*, pp. 196–201, (2000).
- [2] C. Blake and C. Merz. UCI repository of ML databases, 1998.
- [3] A.M. Kaptein, *Meta-Classifier Approach to Reliable Text Classification*, Master’s thesis, Maastricht University, The Netherlands, 2005.
- [4] T.M. Mitchell, *Machine learning*, McGraw-Hill, New York, NY, 1997.
- [5] K. Proedrou, I. Nouretdinov, V. Vovk, and A. Gammerman, ‘Transductive confidence machines for pattern recognition’, in *Proceedings of the 13th European Conference on Machine Learning*, pp. 381–390, (2002).
- [6] C. Saunders, A. Gammerman, and V. Vovk, ‘Transduction with confidence and credibility’, in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 722–726, (1999).
- [7] S. Vanderlooy, I.G. Sprinkhuizen-Kuyper, and E.N. Smirnov, ‘Reliable classifiers in ROC space’, in *Proceedings of the 15th BENELARN Machine Learning Conference*, pp. 27–36, (2006).
- [8] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*, Springer, New York, NY, 2005.

Text Sampling and Re-sampling for Imbalanced Authorship Identification Cases

Efstathios Stamatatos¹

Abstract. Authorship identification can be seen as a single-label multi-class text categorization problem. Very often, there are extremely few training texts at least for some of the candidate authors. In this paper, we present methods to handle imbalanced multi-class textual datasets. The main idea is to segment the training texts into sub-samples according to the size of the class. Hence, minority classes can be segmented into many short samples and majority classes into less and longer samples. Moreover, we explore text re-sampling in order to construct a training set according to a desirable distribution over the classes. Essentially, text re-sampling can be viewed as providing new synthetic data that increase the training size of a class. Based on a corpus of newswire stories in English we present authorship identification experiments on various multi-class imbalanced cases.

1 INTRODUCTION

In recent years, researchers have paid increasing attention to authorship analysis in the framework of practical applications, such as verifying the authorship of emails and electronic messages, plagiarism detection in student essays, and forensic cases [1]. Authorship identification can be seen as a single-label multi-class text categorization problem where the candidate authors play the role of the classes. As concerns the text representation, various measures have been proposed in order to quantify the stylistic choices of the authors including function word frequencies, character n -gram frequencies, vocabulary richness measures, word-class frequencies, and syntactic analysis measures [2].

Very often, a common problem in authorship identification is the lack of sufficient text samples of undisputed authorship (at least for some of the candidate authors). On the other hand, a big amount of text samples may be available for other candidate authors. From a machine learning point of view, this constitutes the *class imbalance* problem (i.e., uneven distribution of the training set over the classes). This problem has been studied mainly within the framework of two-class datasets. The main approaches to deal with class imbalance attempt to re-balance the training set by performing either under-sampling of the majority class or over-sampling of the minority class. In general, the former approach has been proved to work better [3]. Alternatively, the sensitivity of the classification algorithm can be modified so that errors on minority class to be costlier than errors on majority class. Last but not least, the SMOTE approach creates new synthetic training data for the minority class [4]. This is achieved by adding a small value to some of the features of original training data and producing new data which lie close to

the original ones in the multi-dimensional space of the problem.

Given a text categorization task, each training text is considered as a unit for constructing the training set. Usually, the length of the training texts is fixed or defined by the source of the documents. Moreover, text representations usually produce sparse data not quite suitable for a SMOTE-like approach. In this paper, we present methods to efficiently segment the training texts into sub-samples according to the size of the class. Textual data can be handled flexibly so that to produce a variable amount of text samples of variable length. Moreover, by using text re-sampling methods it is possible to provide new synthetic data.

2 AUTHORSHIP IDENTIFICATION

In this paper, we are based on the frequencies of character n -grams for text representation. Let $\mathbf{G}_d = \{g_1, g_2, \dots, g_d\}$ be the ordered set (by decreasing frequency) of the most frequent character n -grams of the training set. Consider f_{ij} as the normalized frequency of the j -th n -gram of \mathbf{G}_d in the i -th text. Then, a text x_i is represented as the vector $\langle f_{i1}, f_{i2}, \dots, f_{id} \rangle$ (in this study, $d=5,000$ and $n=3$). A SVM, a model able to handle highly dimensional and sparse data, is then applied to these vectors. The Weka implementation was used with default parameters. Note that this approach is language-independent. However, for achieving best results given a particular corpus or natural language, one should explore the most appropriate amount and length of character n -grams.

An English corpus of newswire stories by 10 authors (100 texts per author) taken from the new Reuters Corpus Volume 1 has been used in this study. Apart from belonging to the same genre, all the texts fall under the CCAT topic (corporate/industrial) in order to minimize the factors that distinguish the classes despite authorship. Initially, this corpus was divided into non-overlapping training and test texts of equal size. In order to simulate the imbalance conditions of a real-world authorship identification case, a Gaussian distribution of training texts over the candidate authors is assumed. Given this setting, the *multi-class imbalance ratio* of the problem can be defined as *peak/base*, where *peak* and *base* are the size (in training texts) of the biggest and smallest classes, respectively. Figure 1 (left) shows an example distribution of the training set for *peak*=20 and *base*=5. Authors near the center of the distribution are the majority classes while the authors at both sides of the distribution are the minority classes. Different values of *base* and *peak* produce multi-class imbalanced datasets (see Table 1).

3 TESTED METHODS

When using all the training texts, the accuracy is 79.4% which is considered as the upper bound. On the other hand, when each training text is considered as unit and all training texts are used, a

¹ Dept. of Information and Communication Systems Eng., University of the Aegean, 83200, Karlovassi, Greece, email: stamatatos@aegean.gr

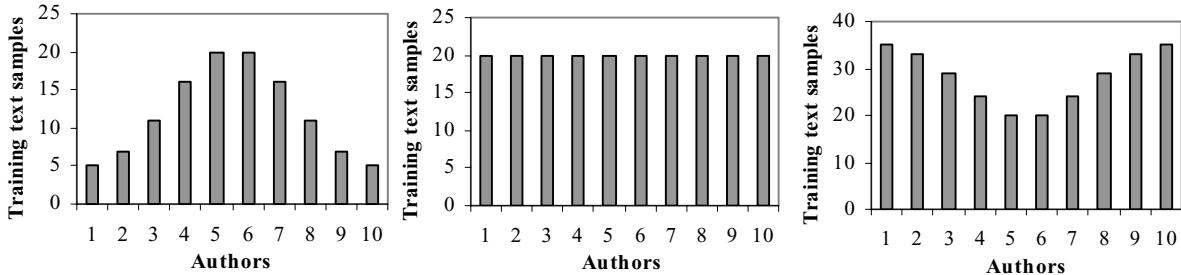


Figure 1. Distribution of text samples over the authors in multi-class training sets. Left: original imbalanced training text samples. Middle: Balanced training text samples produced by method-3. Right: Imbalanced training text samples produced by method-4.

Table 1. Accuracy results for the presented methods on different imbalanced cases together with upper (UB) and lower bounds (LB).

Base	Peak	Ratio	Accuracy (%)				
			UB	LB	M1	M2	M3
2	10	5	79.4	51.2	52.4	56.8	56.6 57.34
2	20	10	79.4	47.4	52.4	56.8	51.8 55.38
2	50	25	79.4	53.6	52.4	56.8	54.6 59.02
5	10	2	79.4	60	49.8	59.8	65.2 58.52
5	20	4	79.4	56.4	49.8	59.8	59 61.44
5	50	10	79.4	59.6	49.8	59.8	62.4 65.76
10	20	2	79.4	62.8	61.4	68.8	65.6 66.18
10	50	5	79.4	65	61.4	68.8	68 69.18

lower bound of accuracy is provided for any imbalanced case. The following methods were tested (results in Table 1):

- Method-1: Under-sampling of the majority classes. For all authors, exactly *base* training texts were used.
- Method-2: Under-sampling of the majority classes using fixed length samples. All the available training texts were concatenated in one text per author. Let x_{min} be the size (in text lines) of the shortest author file. Then, the first x_{min} text lines of each author file were segmented into text sub-samples of length a (in text lines). It was observed that small values of a (2 or 3) tend to provide better results (in Table 1, $a=3$). Note that, in RCV1 each text line comprises one full sentence.
- Method-3: Re-balancing the dataset by variable length text samples. As previously, author files are produced. Let x_i and x_{max} be the text length (in text lines) of the i -th author and the longest author file, respectively. Then, each author file is segmented into text sub-samples of length x_{max}/x_i . Thus, a balanced dataset is produced having k text samples per class (in Table 1, $k=50$).
- Method-4: Re-balancing the dataset by text re-sampling. Again, author files are produced. A variable number of text samples per author is produced according to the length of their file. Many short text samples are produced for the minority classes and less but longer text samples are produced for the majority classes. The text lines included in a text sample are selected randomly and a text line may be included in more than one text sample (in Table 1, average accuracy after 50 runs is shown). Note that this method produces an imbalanced training set (in contrast to methods 1-3). However, the originally minority classes are now represented by more samples than the originally majority classes (see Figure 1).

Table 2 shows the identification accuracy per author (when *base*=5 and *peak*=20) as deviation from the baseline. The baseline method roughly resembles the distribution of training texts over the authors (the more training texts of one author, the better the results). Method-1 improves minority authors but fails to keep the majority authors on high level. Method-2 improves the minority authors

Table 2. The identification accuracy per author of the presented methods on imbalanced corpus (*base*=5, *peak*=20) expressed as deviation from the lower bound (LB).

Author	Tr. Set	LB	M1	M2	M3	M4
A01	5	36	+20	+36	+10	+18
A02	7	26	+22	+28	-16	+12
A03	11	66	-22	-30	-4	-18
A04	16	38	-16	+34	+6	-4
A05	20	100	-12	-8	0	0
A06	20	100	-18	-4	0	-8
A07	16	98	-74	-36	0	-10
A08	11	56	-26	-38	+12	-2
A09	7	6	+12	+10	0	+12
A10	5	38	+48	+42	+18	+46
Accuracy		56.4	-6.6	+3.4	+2.6	+4.6

without a dramatic loss in majority authors. Method-3 achieves to keep majority authors on high level (it even improves them) but it fails to significantly improve the minority authors. Finally, method-4 considerably improves minority authors with the cost of a slight reduction on the majority authors.

4 CONCLUSION

Since textual data can be easily segmented in small pieces, they can be handled more flexibly in comparison to other kinds of data. Various text sampling and re-sampling methods were examined to handle multi-class imbalanced textual data. The main idea of the most successful method was to produce many short text samples for the minority classes and less but longer text samples for the majority classes. The basic methods presented here can be combined in order to further improve the results. For instance, method-3 and method-4 can be applied together. Alternatively, they could be used to train different classifiers which, then, can be combined in an ensemble scheme. Recall from Table 2 that the classification errors made by these methods are to a great extent uncorrelated, a crucial condition to build effective ensembles.

REFERENCES

- [1] A. Abbasi and H. Chen, ‘Applying Authorship Analysis to Extremist-Group Web Forum Messages’ *IEEE Intelligent Systems*, **20**(5), 67-75, (2005).
- [2] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, ‘Automatic Text Categorization in Terms of Genre and Author’, *Computational Linguistics*, **26**(4), 471-495, (2000).
- [3] N. Japkowicz, and S. Stephen, ‘The Class Imbalance Problem: A Systematic Study’, *Intelligent Data Analysis*, **6**(5), 429-450, (2002).
- [4] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, ‘SMOTE: Synthetic Minority Over-sampling Technique’. *Journal of Artificial Intelligence Research*, **16**, 321-357, (2002).

Is Web Genre Identification Feasible?

Benno Stein¹ and Sven Meyer zu Eissen¹

Abstract. This paper contributes to a facet from the area of Web Information Retrieval that has recently received much attention: The satisfaction of a user's personal information need with respect to text type, presentation type, or information quality. We imply that such properties can be quantified for all kinds of Web documents, and we subsume them under the term "Web genre" or "genre".

Recent surveys show that there is—to a certain degree—a common understanding of Web genre. However, the strictness by which genre and non-genre aspects of a document are experienced is an individual matter. To get a better understanding of the challenges of Web genre identification and its possible limits we investigate in this paper a very interesting question, which has not been posed by now:

Given a categorization \mathcal{C} of documents (or bookmarks, links, document identifiers), can we provide a reliable assessment whether \mathcal{C} is governed by topic or by genre considerations?

Keywords unsupervised learning, knowledge discovery, text mining, personal information retrieval

1 INTRODUCTION

Nearly all retrieval processes are topic-centered: We type in a keyword, provide a sample document, or browse a directory tree to get the desired piece of information. However, with the number of indexed documents develops the urgent need for information quality: Users are interested in certain *kinds* of information, or, as it is called here, in particular genres. In connection with text documents genre describes, among others, the set of conventions in the way in which information is presented, such as the style of writing, the presentation style, or the functional trait. An in-depth discussion of the term genre is beyond the scope of this paper,² but, for the time being it is sufficient to remember the following characterization: Topic and genre are orthogonal—or, with Dewdney [3]: "The form is the substance." This is illustrated in Figure 1, where the same articles of a newspaper page are classified under both topic and genre considerations.

Genre *identification* shall discover groups of texts that share a common form of transmission, purpose, or discourse properties [10, 12]. This means in the WWW context that genre identification can be understood as differentiation between research Web pages, personal experience reports, or commercial product information, for example. The application scenario of our paper connects at this point: Given a user's categorized document collection, \mathcal{C} (in the form of bookmark folders for instance), we ask whether one is able to reliably determine the organization principle, say, the underlying categorization type behind \mathcal{C} : Is it topic or genre?

The remainder of this paper outlines Web genre research, provides technical background, and answers the question posed.



Figure 1. The difference between topic and genre, illustrated at a newspaper page.

1.1 Related Work on Web Genre

There is little work on automatic Web genre identification, and, the question of feasibility is—if at all—answered indirectly only, following a simplistic three step approach: (i) definition of particular genre classes, (ii) compilation of a respective genre corpus, (iii) quantification of the learnability by constructing a classifier. In the following we organize the research in an ascending chronological order.

Bretan et al. propose a richer representation of retrieval results in Web search interfaces. Their approach combines content-based clustering and genre-based classification that employs simple part-of-speech information along with substantial text statistics. The features are processed with the C4.5 algorithm; the authors give no information about the achieved classification performance [2]. Based on an exploratory user study Roussinov et al. develop a genre scheme that comprises five genre classes: Topic, Publication, Product, Education, FAQ. Their work describes an ongoing study, and no discovery approach has been implemented [9]. Dimitrova et al. argue that shallow text classification techniques can be used to sort documents according to genre. The paper describes an ongoing study but experience related to classification performance is not reported [4]. Lee and Myaeng define seven genre types. Aside from Web-specific genres like Q&A or Homepage, the authors use also the newspaper-specific genres Reportage and Editorial. The feature set is a list of about hundred document terms tailored to the different genre classes [7]. Meyer zu Eissen and Stein report on a user study on Web genre usefulness from which they derive eight genre classes, which in turn form the building blocks of three genre profiles: Education, Geek, and Private. Within their comprehensive experiments classification performances between 60% and 80% were achieved [8]. Boese investigates the effects of Web document evolution on genre classification and poses the question: "How much do Web pages change over time within each genre?" The author answers this and related questions for two publicly available genre corpora [1].

Note that existing classification approaches treat not the problem that is addressed here; they start with the assumption that the type of the analyzed collection is *a-priori* known, namely, genre.

¹ Faculty of Media / Media Systems. Bauhaus University Weimar, Germany.
benno.stein@medien.uni-weimar.de

² Santini has compiled an up-to-date discussion of this term [10].

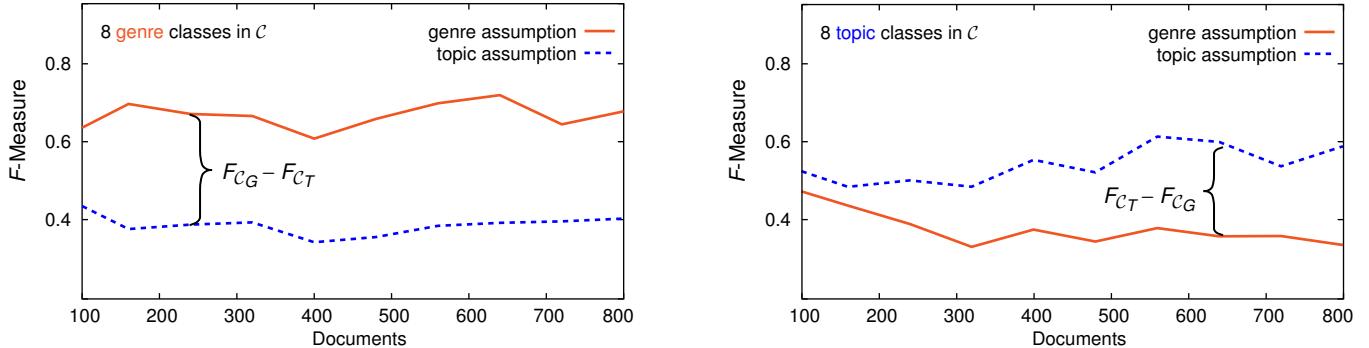


Figure 2. Plots that quantify the adequacy of the document models R_G and R_T . They unveil whether a categorization \mathcal{C} is organized by genre or by topic.

2 DETERMINANTS OF A GENRE CLASSIFIER

With respect to the investigated features the existing approaches to genre identification fall into three groups: Classifiers that rely on a subset of a document's terms [11, 7], classifiers that employ linguistic features along with additional features related to text statistics and computational linguistics [6, 8], or both [5]. The following list gives an overview over the different feature types:

- customariness and style features
- part-of-speech and syntactic group analysis
- closed-class word sets and presentation-related features

Based on these features a powerful document retrieval model, R_G , for genre identification can be built. To keep the computational footprint of our genre model small we applied a discriminant analysis to select 18 features along with an appropriate weighting scheme.

By contrast, to capture the gist of a document with respect to its *topic*, the vector space model, R_T , is the most successful document retrieval model. It encodes a document d as a simple vector, which comprises weighted frequency values of the terms occurring in d .

3 HYPOTHESIZING CATEGORIZATION TYPES

Let D be a set of documents. An exclusive categorization $\mathcal{C} \subseteq \{C_1, \dots, C_k\} | C \subseteq D$ of D is a division of D into sets for which $\bigcup_{C_i \in \mathcal{C}} C_i = D$, and $\forall C_i, C_j \in \mathcal{C} : C_i \cap C_{j \neq i} = \emptyset$. The categorization \mathcal{C} may be governed by topic or by genre considerations, and we introduce the following procedure to verify the underlying categorization type:

1. Construct for each $d \in D$ two document models, one under the genre document retrieval model, R_G , and one under the topic document retrieval model, R_T .
2. Based on a similarity measure (Euclidean or cos-similarity) construct two similarity graphs G_G and G_T . The edge weights in these graphs result from the similarity computations under R_G and R_T respectively.
3. Apply a clustering algorithm to the graphs G_G and G_T . The resulting clusterings are designated as \mathcal{C}_G and \mathcal{C}_T .
4. Compute the F -measure (or another external reference measure) to quantify the congruence between \mathcal{C} and \mathcal{C}_G as well as between \mathcal{C} and \mathcal{C}_T . The resulting values are designated as F_{C_G} and F_{C_T} .
5. If $|F_{C_G} - F_{C_T}|$ is significant, \mathcal{C} is organized under genre considerations if $F_{C_G} > F_{C_T}$, and under topic considerations otherwise.

The F -measure quantifies the degree of congruence between a (human) reference categorization $\mathcal{C} = \{C_1, \dots, C_k\}$ and a clustering $\mathcal{C}' = \{C'_1, \dots, C'_l\}$. The recall of cluster j with respect to category i , $rec(i, j)$, is defined as $|C'_j \cap C_i| / |C_i|$. The precision of cluster j with respect to category i , $prec(i, j)$, is defined as $|C'_j \cap C_i| / |C'_j|$. The F -measure of a clustering \mathcal{C}' , $F_{\mathcal{C}'}$, is:

$$F_{\mathcal{C}'} = \sum_{i=1}^k \frac{|C_i|}{|D|} \cdot \max_{j=1, \dots, l} \{F_{i,j}\}, \text{ with } F_{i,j} = \frac{2 \cdot prec(i, j) \cdot rec(i, j)}{prec(i, j) + rec(i, j)}$$

A perfect clustering matches the given categories exactly and yields an F -measure value of 1.

Experiments Our experiments rely on the corpus of [8], where eight Web genre classes are distinguished: Help, Article, Discussion, Shop, Portrayal (priv and non-priv), Link Collection, and Download. The orthogonal topic categorization distinguishes the following eight topics: Sports, Annual results, International relations, Religion, Crime, Management moves, Money supply, Legal/judicial.

Based on this corpus we compiled 40 categorizations of different sizes and under both topic and genre considerations. It turned out that for each of these categorizations its type could be unambiguously determined by computing $|F_{C_G} - F_{C_T}|$, whereas a genre model comparable to [2] and as topic model the vector space model was employed. Figure 2 shows the developing of the respective F -measure values F_{C_G} and F_{C_T} . The runtime complexity is dominated by the cluster analysis, and, using k -means, linear in \mathcal{C} .

Our current research focuses on document models for special retrieval situations and related learning strategies.

REFERENCES

- [1] E. S. Boese and A. E. Howe, 'Effects of Web Document Evolution on Genre Classification', in *Proc. CIKM'05*, (2005).
- [2] I. Bretan, J. Dewe, A. Hallberg, and N. Wolkert. Web-specific genre visualization, (1999).
- [3] N. Dewdney, C. VanEss-Dykema, and R. MacMillan, 'The form is the substance: Classification of genres in text', in *Proc. ACL*, (2001).
- [4] M. Dimitrova, A. Finn, N. Kushmerick, and B. Smyth, 'Web genre visualization', in *Proc. Human Factors in Comp. Systems*, (2002).
- [5] A. Finn and N. Kushmerick, 'Learning to Classify Documents According to Genre', in *IJCAI-03 Workshop on Computational Approaches to Style Analysis and Synthesis*, (2003).
- [6] B. Kessler, G. Nunberg, and H. Schütze, 'Automatic detection of text genre', in *Proc. ACL and EACL*, Somerset, New Jersey, (1997).
- [7] Y.-B. Lee and S. . Myaeng, 'Text genre classification with genre-revealing and subject-revealing features', in *Proc. SIGIR*, (2002).
- [8] S. Meyer zu Eissen and B. Stein, 'Genre Classification of Web Pages: User Study and Feasibility Analysis', in *KI 2004: Advances in Artificial Intelligence*, volume 3228 LNAI, Springer, (2004).
- [9] D. Roussinov, K. Crowston, M. Nilan, B. Kwasnik, J. Cai, and X. Liu, 'Genre based navigation on the web', in *Proc. 34th Hawaii International Conference on System Sciences*, (2001).
- [10] M. Santini, 'State-of-the-Art on Automatic Genre Identification', Technical report, ITRI, University of Brighton, UK, (2004).
- [11] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, 'Text genre detection using common word frequencies', in *Proc. Conf. on Computational Linguistics*, Saarbrücken, Germany, (2000).
- [12] J. Swales, *Genre Analysis. English in Academic and Research Settings*, Cambridge University Press, 1990.

7. Natural Language Processing

This page intentionally left blank

Adaptive Context-based term (re)weighting

An experiment on Single-Word Question Answering

Marco Ernandes and Giovanni Angelini and Marco Gori and Leonardo Rigutini and Franco Scarselli¹

Abstract. Term weighting is a crucial task in many Information Retrieval applications. Common approaches are based either on statistical or on natural language analysis. In this paper, we present a new algorithm that capitalizes from the advantages of both the strategies. In the proposed method, the weights are computed by a parametric function, called *Context Function*, that models the semantic influence exercised amongst the terms. The Context Function is learned by examples, so that its implementation is mostly automatic. The algorithm was successfully tested on a data set of crossword clues, which represent a case of Single-Word Question Answering.

1 Introduction

Term weighting is an important task in many areas of Text Processing, including, Document Retrieval, Text Categorization and Question Answering (QA). The goal of term weighting is to assign to each term w found in a collection of text documents a specific score $s(w)$ that measures the importance, with respect to a certain goal, of the information represented by the word. Common approaches to term weighting can be divided into two groups: statistical and linguistic techniques. Statistical techniques [1] (e.g. TFIDF) are efficient and easy to develop, but they tend to consider the words of a document as unordered and independent. The techniques inspired by natural language theories [6], as morphological analysis, naturally exploit the information provided by word contexts. This makes the processing more expressive, but also slower and more complex to design.

In this paper, we present a term weighting algorithm that aims to combine the advantages of both statistical and linguistic strategies. The method exploits the relationships among the words of a document. The intuition is that the relevance of a term can be computed recursively as the combination of its intrinsic relevance and the relevance of the terms that appear within the same context. The influence exercised by a word on another one is computed using a parametric function, called *Context Function*. This function can use both statistical and linguistic information, and it can be trained by examples.

The Context-based algorithm has been evaluated on a specific problem, that of Single-Word Question Answering (QA), where the goal is to find the single correct word that answers a given question. The experimental results prove that the approach is viable.

2 Adaptive Context-based term (re)weighting

The proposed method exploits the word contexts. A word context primarily consists of the text surrounding a given word, but could also include other features, e.g. document titles, hyper-linked documents. The basic idea is that, in order to measure the relevance of a word with respect to a certain goal (e.g. a query, a document category), the features of the context in which the term appears are important as

well as the features of the word itself. In this work we assume that a text document can be represented by a social network [5], where the importance of the words can be computed on the basis of their neighbours. More precisely, the weight $s(w)$ of the word w is computed as

$$s(w) = (1 - \lambda)d_w + \lambda \sum_{u \in r(w)} s(u)c_{w,u}, \quad (1)$$

where d_w is the *default score* of w , $r(w)$ is the set of words that belong to the context of w , $c_{w,u}$ is a real number measuring the influence exercised by u over w , and $\lambda \in [0, 1]$ is a damping factor.

Eq. (1) defines the term weights by a sparse linear system of equations. In our experiments, the solution of such a system was computed by Jacobi algorithm, an efficient algorithm which can be applied even on huge problems with billions of variables [3].

Context Functions In order to define the influence factors, it has to be taken into account that words are multiply instantiated in different positions of a document, and each instance (a word-occurrence) is affected by a different context. Therefore, we distinguish between words, w , and word occurrences, \hat{w} . We assume that $c_{w,u}$ can be computed as the sum of the contributions of all the occurrences \hat{w} , \hat{u} of w and u , respectively, such that \hat{u} belongs to the context of \hat{w}

$$c_{w,u} = \sum_{\hat{w} \in occ(w)} \sum_{\hat{u} \in ict(\hat{w}, u)} C_p(\hat{w}, \hat{u}). \quad (2)$$

Here, $occ(w)$ is the set of instances of word w , and $ict(\hat{w}, u)$ is the set of the occurrences of u that belong to the context of \hat{w} , i.e. $ict(\hat{w}, u) = occ(u) \cap ctxt(\hat{w})$, where $ctxt(\hat{w})$ is the context of \hat{w} ; p is a set of parameters, and $C_p(\hat{w}, \hat{u})$ is a parametric function that establishes the strength of the influence between the instances \hat{w} (the word under evaluation) and \hat{u} (the context word). In this work we define the context of a word $ctxt(\hat{w})$ as the set of words that are contained in the same document and within the surround of \hat{w} .

The function $C_p(\hat{w}, \hat{u})$ establishes how word couples can influence one another on the basis of features extracted from the words and from the relationships between words. This function can exploit any sort of feature from \hat{w} and \hat{u} : info-theoretical, morphological or lexical. The features that have been used in our preliminary experiments are exclusively statistical (Tab. 1).

The most general approach to the implementation of $C_p(\hat{w}, \hat{u})$ is by a modeling tool that has the universal approximation property, e.g. neural networks, polynomials, and rationales. For the introductory scope of this paper we preferred to adopt a simpler implementation of $C_p(\hat{w}, \hat{u})$. We defined the influence function as $C_p(\hat{w}, \hat{u}) = \prod_{i=0}^n \sigma(\alpha_i x_i + \beta_i)$, where x_i is the value associated with the i -th feature, α_i, β_i are the model parameters, and σ is the logistic sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. Each term $\sigma(\alpha_i x_i + \beta_i)$ is a sort of soft switch related to the i -th feature and controlled by α_i (steepness and direction) and β_i (medium value) so that the whole function is a sort of boolean expression composed by AND operators.

¹ Dip. di Ingegneria dell'Informazione, Università di Siena, via Roma 56, 53100 - Siena - Italy, email: {ernandes, angelini, marco, rigutini, franco}@dii.unisi.it

Name	Feature Set
FS-A	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$
FS-B	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$, $dist(\hat{u}, Q)$
FS-B*	$idf(w)$, $idf(u)$, $dist(\hat{w}, \hat{u})$, $dist(\hat{u}, Q)$, <i>sw-list</i>

Table 1. The set of features used by the Context Functions. $dist(\hat{w}, \hat{u})$ is the number of words between \hat{w} and \hat{u} , $dist(\hat{u}, Q)$ is the number words between \hat{u} and the word occurrences of query Q . Symbol *sw-list* denotes the cases that a stop word list is used to remove non informative terms.

Learning Context Functions The goal of the learning procedure is the optimization of a cost function e_p that measures the performance of the weighting system. We adopted the resilient parameter adaptation [4], which is an efficient technique that updates the parameters using the signs of the partial derivatives of the cost function.

The cost function e_p depends on the specific task that is addressed by the application. The mean reciprocal rank of the correct answer (MRR) is a standard positional and cumulative evaluation measure can be used for QA evaluation. Formally, given a set of queries $Q = \{q_1, \dots, q_n\}$ and denoted by $pos(a^q)$ the position of the correct answer a^q for question q , we have $MRR(Q) = \frac{1}{n} \sum_{q=1}^n \frac{1}{pos(a^q)}$.

On the other hand, MRR is a discrete function and cannot be used for gradient-based learning procedures. In order to design a differentiable cost function, we defined a “soft position” function *soft_pos* to replaces *pos*: $soft_pos(a) = \sum_{w \in W, w \neq a} \sigma(\gamma(s(w) - s(a)))$, where W is the set of words considered by the weighting algorithm, a is the correct answer, γ is a predefined parameter and σ is a sigmoid function. In fact, it can be easily observed that for large γ , *soft_pos(a)* approximates *pos(a)*. Finally, we defined a differentiable evaluation function which approximates MRR simply by replacing *pos(a)* with *soft_pos(a)*. Once the system is trained using this soft function, the results can be measured by the standard MRR.

3 Experimental Results

The Single-Word QA problem The proposed method was experimentally assessed on Single-Word QA, a special case of QA where each question has to be answered with a single correct word, given a number of documents related to the question. A popular and challenging example of Single-Word QA is provided by crossword clues.

Our experiments aim to show that the Context-based algorithm improves the ranking quality of the candidate answers provided by WebCrow, a Web-based system designed to answer crossword clues [2].

The dataset consisted of 525 Italian crossword clues randomly extracted from the archive in [2]. The examples were divided into two subsets: *NE*-questions (165 examples), whose answers are factoid Named Entities, and, *nonNE*-questions (360 examples), whose answers are non-Named Entities (common nouns, adjectives, verbs, and so on. e.g. *<Reduce the area by two: foldinhalf>*).

Compared approaches The Context-based algorithm was compared with three different weighting techniques. The first is TFIDF, a standard measure that gives a statistical evaluation of the importance of a word to a document. The other two techniques, *WebCrow-S* and *WebCrow-SM*, are two ranking methods used in WebCrow. WebCrow-S exploits only statistical information of the words, the documents and the queries. WebCrow-SM additionally uses a morphological analysis of the documents joined with a morphological Answer Type classification of the clues (see [2]).

The experiments The subsets (*NE* and *nonNE*-questions) were randomly divided into a train set (40%) and test set (60%). Each example consisted of a question/answer couple, a set of H documents related by Google to the question; the vector of the ranked terms \mathbf{W} (the candidate answers), extracted from the H documents. Several

	Def TW	FS	NE	nonNE
TFIDF			0,216	0,071
TFIDF + context reweighting	FS-A	0,233	-	
WebCrow-S			0,290	-
WebCrow-S + context reweighting	FS-B	0,346	-	
WebCrow-S + context reweighting	FS-B*	0,355	-	
WebCrow-SM			0,293	0,104
WebCrow-SM + context reweighting	FS-B*	0,345	0,121	

Table 2. MRR performance. The results of the three baseline algorithms are given in bold. Def TW denotes the algorithm used for the default term weights d_w , FS specifies the Feature Set of the Context Function.

Context Functions with different set of features (Tab. 1) were trained. We chose 20 as a fixed dimension for the context window.

The achieved results, displayed in Table 2, confirm that the Context-based algorithm outperformed the other weighting techniques. For the *NE* subset (Table 2, *NE* column), the performance was impressive. In particular, the reweighting of the default scores provided by WebCrow-S, produced a 22,3% increment of the MRR (from 0,29 to 0,355, row 5). An insightful resume of the performance improvements is provided by the Success Rate curve (Fig. 1), that shows the probability of finding the correct answer within the first N words of \mathbf{W} . Each weighting scheme under comparison appears consistently below the curve of its Context-based reweighting.

4 Conclusions and Further work

In this paper we proposed a novel term weighting that exploits the information provided by word contexts. The approach has been proved to be very effective on the problem of Single-Word Web-based Question Answering. Future matter of research includes the application of the method to Document Classification and Information Extraction.

ACKNOWLEDGEMENTS

This work has been funded by the Google Research Awards Program. We are grateful to M. Bianchini, E. Di Iorio and G. Monfardini for their important advises and we thank M. Diligenti for his support.

REFERENCES

- [1] M. Berry, Z. Drmac, and E. Jessup, ‘Matrices, vector spaces, and information retrieval’, *SIAM Rev.*, **41**(2), 335–362, (1999).
- [2] M. Ernandes, G. Angelini, and M. Gori, ‘Webcrow: A web-based system for crossword solving.’, in *Proc. of AAAI ’05*, Pittsburgh, USA, (2005).
- [3] L. Page, S. Brin, R. Motwani, and T. Winograd, ‘The pagerank citation ranking: Bringing order to the web’, in *Proc. of ASIS ’98*, Pittsburgh, USA, (1998).
- [4] M. Riedmiller and H. Braun, ‘A direct adaptive method for faster back-propagation learning: The RPROP algorithm’, in *Proc. of ICNN ’93*, San Francisco, USA, (1993).
- [5] J. R. Seeley, ‘The net of reciprocal influence’, *Canadian Journal of Psychology*, **3**(4), 234–240, (1949).
- [6] E. Voorhees, ‘Natural language processing and information retrieval’, in *Proc. of SCIE ’99*, Frascati, Italy, (1999).

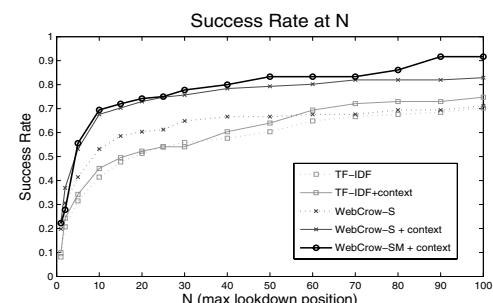


Figure 1. The probability (vertical axis) of finding the correct answer within the first N candidate answers (horizontal axis).

How to Analyze Free Text Descriptions for Recommending TV Programmes?

Bernd Ludwig¹ and Stefan Mandl²

Abstract. This paper presents an approach to exploit free text descriptions of TV programmes as available from EPG data sets for a recommendation system that takes the content of programmes into account³. The paper focusses on classifying free text descriptions in relation to natural language user queries.

1 Introduction

Via satellite the Electronic Programme Guide (EPG) provides an enormous amount of information about TV programmes including natural language descriptions of programmes. Viewers are overwhelmed by the huge number of channels when they select a programme. Therefore, they would prefer to know more about the content of a programme when deciding whether to watch it or not. This requires that a recommendation system has information about the content available and can analyze programme descriptions semantically comparing it with the viewer's interests.

At the beginning of this paper, we compare our work with previous research efforts. In Section 1.2 we report on a user study that shows how viewers select programmes. In Section 2 we explain our approach to analyse free text descriptions. We conclude the paper with a report on a first evaluation of the system in Section 3.

1.1 The State-of-the-Art Recommendation Systems

The design and implementation of TV recommendation systems has attracted great interest in different research groups already. They use sophisticated user models such as in [1]. In order to allow for default reasoning, stereotypes for users are applied which are based on the analysis of the average user's lifestyle (see [4]). In order to increase the user's confidence in the system proposals, the generation of trust-worthy suggestions that take programmes watched earlier into account has been studied in detail (see [2]). In contrast to the reported research, we focus on the question of how a system could take the contents of TV programmes into account.

1.2 How Do Viewers Select TV Programmes?

A user study [7] conducted as part of the research project EMBASSI (see [5, 6]) revealed a number of interesting facts about how users like to select TV programmes. Candidates were situated in front

of a computer display that suggested an automatic recommendation system to be at work. Actually, in a room nearby, a human person monitored the candidate and responded according to the information available from a TV magazine. The users were allowed to ask arbitrary questions about the currently available TV programmes. On the display a list of proposals was presented and users could ask more specific questions on certain proposals – mostly about their *content*:

Liebe, Romantik (Love, romance)

Spannung, Magie, Fantasie (Tension, magic, fantasy)

Eine politische Sendung (A programme on politics)

Users often expressed emotional attitudes they desired the programme to have:

Entspannen (To relax)

Show, Witz (Show, fun)

2 Recommendations Based on Free Texts

Recommendations for TV programmes could be improved if they relied not on genre types only, but if it was possible to know more about the content of programmes and connect this information with its knowledge about the user and his/her inquiry in natural language. The current section explains our solution to this issue.

The method we propose for shallow semantic analysis of EPG programme descriptions is based on the DORNSEIFF lexicon for German. Like a thesaurus, it groups words according to certain topics, i.e. in each group there are words (even of different word categories) that describe a particular aspect of a certain topic. The DORNSEIFF lexicon⁵ is not a synonym lexicon, but a “topic” lexicon. Structured in a two-level hierarchy, the lexicon organizes topics in chapters (e.g. chapter 15 contains subtopics *social life*) and subchapters (e.g. subchapter 15.39 is the topic *reward*). If the meaning of a word is ambiguous, it is listed in more than one subchapter.

The central idea is to quantify on the one hand how well a free text for a TV programme (its content) matches the user query and on the other hand whether the emotions entailed in the query can be found in the free text as well. Both aspects cover almost all user queries we have collected (see Sect. 1.2). Proposals are computed by finding pareto-optimal sets for both criteria.

2.1 Measuring the Semantic Distance

The first step is the identification of those components of the feature vector that cover the “main theme” of a programme description. A

¹ Chair for Artificial Intelligence, University of Erlangen-Nuremberg, Germany, email: Bernd.Ludwig@informatik.uni-erlangen.de

² Chair for Artificial Intelligence, University of Erlangen-Nuremberg, Germany, email: Stefan.Mandl@informatik.uni-erlangen.de

³ Our research is sponsored by *Software-Offensive Bayern*⁴. Ferdinand Herrmann, Heike Ott, Kristina Makedonska, and Sebastian von Mammen provided valuable help implementing major parts of the presented system.

⁵ Interested readers can test the online version of the DORNSEIFF lexicon on <http://wortschatz.uni-leipzig.de>. It contains about 9 million different words and word groups.

triangular semantic-distance matrix is computed for the DORNSEIFF groups occurring at least once. Each group name (a German noun, adjective, or verb) is looked up in GERMANET. We get a set of trees of the group name's synsets. The distance $\text{dist}(s, t)$ between group names s and t is measured by the number of steps from the leaf to the first node in the tree for s that is also found in the tree for t . If there are multiple readings in GERMANET, the maximum number of steps is taken. If two group names don't have a common hyperonym, the pair is omitted further on.

Next, a mean value for the distance of group names is calculated from the entries in the matrix. Pairs whose distance exceeds the variance are set to have ∞ distance. Finally, group names with no distance smaller than ∞ to any other group name are deleted. The group names remaining after this process are those topics in a free text description which are semantically as close as possible.

The relevance $\text{semrel}(s, t)$ of each pair (s, t) of the remaining DORNSEIFF groups is computed by:

$$\text{semrel}(s, t) = -\log \left(\frac{\text{freq}(s) \cdot \text{freq}(t)}{\text{maxfreq}^2} \cdot \frac{\text{dist}(s, t)}{\text{maxdist}} \right)$$

maxfreq is the highest frequency in the description and maxdist is the greatest distance in the distance matrix.

2.2 Valence and Arousal

For quantifying emotions, we were looking for an exhaustive list of emotions and a mapping from words to basic emotions. COWIE and his colleagues (see [3]) provide such a list of 107 emotional attitudes. We apply this list as follows: In a preprocessing step, we build a mapping for DORNSEIFF groups to \mathbb{R}^2 . For example, the attitude *adventurous* is mapped as follows: In German, *adventurous* means *abenteuerlich*. Looking up the word in the DORNSEIFF lexicon, one finds it in the groups 9.72 *Gefahr* (*danger*), 10.23 *Lächerlich* (*ridiculous*), 10.38 *Tollkühn* (*daredevil*), and 11.26 *Einbildung*, *Wahn* (*illusion*). These groups cover all connotations of the attitude *adventurous* that are expressed in German by *abenteuerlich*.

[3] assigns a position in a two-dimensional diagram to each attitude (for *adventurous* the coordinates are (4.2, 5.9)). The first coordinate expresses the attitude's *valence*, which addresses its quality (ranging from very negative over neutral to very positive). The second coordinate stands for *arousal*, which refers to the (quantitative) activation level of the feeling (ranging from very low to very high). To map words to basic emotions, each DORNSEIFF group is mapped to the position of the related attitude ((4.2, 5.9) in our example). This procedure has been applied to all 107 emotional terms in [3].

For feature extraction, the DORNSEIFF group names are mapped onto a corresponding set of emotions. For each emotion (x, y) out of this set its relevance is computed taking the frequency of its underlying DORNSEIFF group names, the semantic distance to the other groups, and their user rating into account:

$$\text{emrel}(x, y) = -\sum_{s \text{ mapped to } (x, y)} \log \left(\frac{\text{freq}(s)}{\text{maxfreq}} \cdot (1 - \text{ur}(s)) \prod_t \frac{\text{dist}(s, t)}{\text{maxdist}} \right)$$

$\text{ur}(s)$ measures the user's long-time preference of s from 0% up to 100%. Finally, for each programme description d we have two characteristic matrices: one for the content and one for the emotional attitudes of the programme description. The same procedure is applied to the user query. The distance between d and q is defined by the mean square error of the characteristic matrices. Then the pareto-optimal set that optimizes both criteria is determined and presented to the user as the set of proposals for his/her query.

3 A First Evaluation

As we want to develop a system that is considered to offer more comfort and flexibility to the users, the most important figure to evaluate is how good do proposals match the user query in the user's view.

For a first evaluation, in a public presentation of the demonstrator system people of different sex, age, education and interest could test the system as long as they wanted to. They used the NL interface by typing in queries such as those in Sect. 1.2. People tested the system for about 15 minutes and entered around five queries in the average. The system responded by presenting a list of the programmes on air at the time of the query sorted according to how well each programme matched the query. Then the users filled in a questionnaire. 60 questionnaires have been evaluated. One of the questions was: *How useful do you consider the text input-and-search function of the system?* It was answered as follows:

answer	share	answer	share
very useful	36%	useless	7%
useful	26%	very useless	1%
fair	20%	no idea	10%

These results are promising and provide motivation for further work.

4 Conclusions

Applying the DORNSEIFF lexicon appears to be a successful approach to abstract from a given text. It offers a way to generalization that does not exclusively rely on statistical methods which are the key work horse for shallow processing of unrestricted text. Instead of that, corpus linguistic experience can be incorporated into the analyses which leads to new interesting hybrid algorithms. The evaluation of the user feed back indicates that such an approach is also accepted in typical situations of using text retrieval systems.

REFERENCES

- [1] Liliana Ardissono, Christina Gena, Pietro Torasso, Fabio Bellifemmine, Angelo Difino, and Barbara Negro, 'User modelling and recommendation techniques for personalized electronic program guides', in *Personalized Digital Television – Targeting Programs to Individual Viewers*, eds., Liliana Ardissono, Alfred Kobsa, and Mark T. Maybury, volume 6 of *Human-Computer Interaction Series*, chapter 1, 3–26, Springer, (2004).
- [2] Anna L. Buccak, John Zimmerman, and Kaushal Kurapati, 'Personalization: Improving ease-of-use, trust, and accuracy of a tv show recommender', in *Proceedings of the TV'02 workshop on Personalization in TV*, Malaga (Spain), (2002).
- [3] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J.G. Taylor, 'Emotion recognition in human-computer interaction', *IEEE Signal Processing Magazine*, **18**, 32 – 80, (January 2001).
- [4] Cristina Gena, 'Designing tv viewer stereotypes for an electronic program guide', in *Proc. UM2001 Workshop on Personalization in Future TV (TV01)*, Sonthofen, (July 2001).
- [5] Thorsten Herfet, Thomas Kirste, and Michael Schnaider, 'Embassi – multimodal assistance for infotainment and service infrastructures', *Computers and Graphics*, **25**(4), 581–592, (August 2001).
- [6] Thomas Kirste and Thomas Heider, 'Supporting goal-based interaction with dynamic intelligent environments', in *Proceedings of the 15th European Conference on Artificial Intelligence*, ed., Frank van Harmelen, pp. 22–26, Lyon (France), (July 2002). IOS Press.
- [7] Julia Nitschke and Michael Hellenschmidt, 'Design and evaluation of adaptive assistance for the selection of movies', in *Proceedings of IMC 2003 "Assistance, Mobility, Applications"*, Rostock, (June 2003).

Soft Uncoupling of Markov chains for Permeable Language Distinction: A New Algorithm

Richard Nock¹, Pascal Vaillant¹, Frank Nielsen², Claudia Henry¹

Abstract.

Without prior knowledge, distinguishing different languages may be a hard task, especially when their borders are permeable.

We develop an extension of spectral clustering — a powerful unsupervised classification toolbox — that is shown to resolve accurately the task of soft language distinction. At the heart of our approach, we replace the usual hard membership assignment of spectral clustering by a soft, probabilistic assignment, which also presents the advantage to bypass a well-known complexity bottleneck of the method.

Experiments with a readily available system display the potential of the method, which brings a visually appealing soft distinction of languages that may define altogether a whole corpus.

1 Introduction

Many corpora collected in spontaneous situations (e.g. newsgroup or chatroom archives) are not “clean”: different languages are intermingled in the texts. The different languages can be all the more tricky to distinguish as they share a common vocabulary, which tends to be the case when they are genetically close (this is so with the text corpora we are working on, which mix French and Caribbean French Creole). In this paper we tackle the problem of automatically distinguishing languages whose borders tend to be “permeable”, in their use (they are intermingled in texts) as well as in their systems (they are parent dialects with an important share of common vocabulary). No prior information is available, either about the texts in the corpus or about the languages. We use a new spectral clustering algorithm which amounts to finding a best split of the text-generating Markov chain states set. However, the method we propose does not assign every state into a unique cluster (which interprets as: it does not assign a single unique language to each word); instead, it proposes a probability distribution of each state into the different clusters. The algorithm we present is an algorithm of *soft* spectral clustering. Finally, the information is used to build overlapping language lexica; and, back to the text, to help visualize the linguistically homogeneous segments.

2 Maximum-likelihood Markov chains

In a first step, the data found in the corpus is interpreted as the result of a random walk in a Markov chain. The corpus is analysed as a sequence of n tokens of v distinct word-types in a finite vocabulary \mathcal{V} ($n \geq v$). Let $n_{i,j}$ represent the number of times a token of word i is followed by a token of word j in the corpus ($i, j \in [1, v]$), and n_i the total number of tokens of word i in the corpus. We can build a simple bigram-model Markov chain representing the maximum-likelihood model of the generation process fitting in a chain of v

¹ Université des Antilles et de la Guyane, Schoelcher, Martinique (France)

² Sony CS Labs, Tokyo, Japan

states: every transition from state i to state j is then weighted with the probability of transition from word i to word j , which is estimated to be: $n_{i,j}/n_i$. The $v \times v$ matrix with these coefficients is the bigram transition matrix of the texts of the corpus.

This first matrix generally is not symmetric, since $n_{i,j}$ does not necessarily equal $n_{j,i}$. In order to remove any assumption, even the mildest, about the language used in the corpus, we decide to consider the texts as static objects, taking no account of the writing direction: with this purpose, we define a new matrix W whose coefficients are: $w_{i,j} = (n_{i,j} + n_{j,i})/2$. The $w_{i,j}$ somehow represent a direction-independent weight function between word i and word j . A new transition matrix P is defined with coefficients $p_{i,j} = w_{i,j}/n_i$. Writing D the diagonal matrix whose $\langle i, i \rangle$ coefficients represent the number of occurrences of word i ($d_i = n_i$), we can also simply write: $P = D^{-1}W$. The matrix P so defined will be used from now on. It is the transition matrix of a Markov chain \mathfrak{M} . \mathfrak{M} is the maximum-likelihood bigram model Markov chain, not of the texts in the corpus themselves, but of their weak, direction-independent, counterparts (it is like if every text could be read equally from left to right or right to left, with a random direction at every reading step). P is a real $v \times v$ matrix, valued in $[0, 1]$. It is stochastic, since on every line $\sum_{j=1}^v d_{i,j} = d_i$. It is not necessarily symmetric, but as it is defined as the product of two symmetric matrices, it has v real eigenvalues.

\mathfrak{M} as defined above is irreducible. We can also make the assumption that it is aperiodic. This derives from a clearly mild assumption, namely that it satisfies for a vocabulary large enough, as in this case loops of arbitrary long size tend to appear between words. Irreducibility and aperiodicity imply that \mathfrak{M} is ergodic, i.e. regardless of the initial distribution, it will settle down over time to a single stationary distribution π solution of $P^\top \pi = \pi$, with $\pi_i = n_i/n$ [4].

3 From hard to soft spectral clustering

Let us now state the general problem: let $q > 1$ be some user-fixed integer that represents the number of clusters to find within the state space of chain \mathfrak{M} —i.e. the vocabulary of the corpus.

The ideal objective would be to find a mapping $Z : \mathcal{V} \rightarrow \mathbb{S}^q$, with $\mathbb{S} = \{0, 1\}$, mapping that we represent by a matrix $Z = [z_1, z_2, \dots, z_q] \in \mathbb{S}^{v \times q}$. Under appropriate constraints, the mapping should minimize a *multiway normalized cuts* (MNC) criterion [1, 2, 3, 4, 5, 6, 7]:

$$\begin{aligned} \arg \min_{Z \in \mathbb{S}^{v \times q}} \mu(Z) &= \sum_{k=1}^q \kappa_k(Z)/\alpha_k(Z) , \\ \text{s.t. } Z^\top Z &\text{ positive diagonal} \\ \text{s.t. } \text{tr}(Z^\top Z) &= v , \end{aligned} \quad (1)$$

with $\kappa_k(Z) = \sum_{i,j=1}^v w_{i,j}(z_{i,k} - z_{j,k})^2$ (“surface” of cluster k) and $\alpha_k(Z) = \sum_{i=1}^v z_{i,k}^2 d_i$ (“volume”). Since that does not change the value of $\mu(Z)$, we suppose without loss of generality that $w_{i,i} = 0, \forall 1 \leq i \leq v$. Because of the constraints on Z in (1), it induces a natural hard membership assignment on \mathcal{V} (*i.e.* a partition), as follows:

$$\mathcal{V}_k = \{v_i : z_{i,k} = 1\}, \forall 1 \leq k \leq q. \quad (2)$$

There is one appealing reason why clustering gets better as MNC in (1) is minimized. Suppose we start (at $t = 0$) a random walk with the Markov chain \mathfrak{M} , having transition matrix P , and from its stationary distribution π . Let $[\mathcal{V}_k]_t$ be the event that the Markov chain is in cluster k at time $t \geq 1$. We obtain the following result [4]:

$$\mu(Z) = 2 \sum_{k=1}^q \mathbf{Pr}([\overline{\mathcal{V}}_k]_{t+1} | [\mathcal{V}_k]_t) \quad (3)$$

for the partition defined in eq. (2). Thus, $\mu(Z)$ sums the probabilities of escaping a cluster given that the random walk is located inside the cluster: minimizing $\mu(Z)$ amounts to partitioning \mathcal{V} into “stable” components with respect to \mathfrak{M} .

Unfortunately, the minimization of MNC is NP-Hard, already when $q = 2$ [7]. To approximate this problem, the output is relaxed and the goal rewritten as seek:

$$\begin{aligned} \arg \min_{Y \in \mathbb{R}^{v \times q}} \nu(Y) &= \sum_{k=1}^q \kappa_k(Y), \\ \text{s.t. } Y^\top D Y &= I. \end{aligned} \quad (4)$$

This problem is tractable by a spectral decomposition of \mathfrak{M} (see e.g. [7]), which yields that Y is the set of the q column eigenvectors associated to the smallest eigenvalues of the generalized eigenproblem ($\forall 1 \leq k \leq q$):

$$(D - W)\mathbf{y}_k = \lambda_k D\mathbf{y}_k, \quad (5)$$

and it comes $\nu(Y) = 2 \sum_{k=1}^q \lambda_k$. If we suppose, without loss of generality, that eigenvalues are ordered, $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_q$, then it easily comes $\lambda_1 = 0$, associated to a constant eigenvector \mathbf{y}_1 [7]. People usually discard this first eigenvector, and keep the following ones to compute Z after a heuristic thresholding of Y . The proof that this thresholding is heuristic follows from the fact that if we restrict (4) to thresholded matrices (whose rows come from a set of at most q distinct row vectors), then it becomes equivalent to (1), *i.e.* intractable [1].

Notice however that the spectral relaxation finds the *optimal* solution to (4) in time $O(qv^3)$ (without algorithmic sophistication), from which the heuristic thresholding only aims at recovering a hard membership assignment. Whenever a soft membership assignment is preferable, we show that one can be obtained directly from Y , which is optimal with respect to a criterion similar to (3), while its computation bypasses the complexity bottleneck of hard membership, thus killing two birds in one shot.

For this purpose, define matrix \tilde{Y} from Y as $\tilde{y}_{i,k} = d_i y_{i,k}^2$. Then, we have $\tilde{Y}^\top \mathbf{1} = \mathbf{1}$, *i.e.* each column vector $\tilde{\mathbf{y}}_k$ of \tilde{Y} defines a probability distribution over \mathcal{V} . Since $\tilde{\mathbf{y}}_k$ is associated to principal axis k , it seems natural to define it as the probability to draw v_i given that we are in \mathcal{V}_k , the cluster associated to the axis. Following the notations of eq. (3), we thus let:

$$\tilde{y}_{i,k} = \mathbf{Pr}([v_i]_t | [\mathcal{V}_k]_t) \quad (6)$$

be the probability to pick type v_i , given that we are in cluster k , at time t . This is our soft membership assignment: axes define clusters, and the column vectors of \tilde{Y} define the distributions associated to each cluster. Notice that this provides us with a sound extension of the hard MNC solution (2) for which $\tilde{y}_{i,k}$ equals 1 for a single cluster, and zero for the other clusters ($\forall 1 \leq i \leq v$). We also have more, as this brings a direct and non trivial generalization of (3).

Define matrix $P^{(k)}$ such that $p_{i,j}^{(k)} = (w_{i,j} y_{j,k}) / (d_i y_{i,k})$. $p_{i,j}^{(k)}$ is akin to the difference between the probabilities of reaching respectively \mathcal{V}_k and $\overline{\mathcal{V}}_k$ in j , given that the random walk is located on i ($\forall t \geq 0$): $p_{i,j}^{(k)} = \mathbf{Pr}([v_j \wedge \mathcal{V}_k]_{t+1} | [v_i]_t) - \mathbf{Pr}([v_j \wedge \overline{\mathcal{V}}_k]_{t+1} | [v_i]_t)$.

Provided we make the assumption that reaching a type outside cluster k at time $t + 1$ does not depend on the starting point at time t , an assumption similar to the memoryless property of Markov chains, we obtain our main result, whose proof relies on applications of Bayes rules.

$$\mathbf{Theorem 1} \quad \nu(Y) = 4 \sum_{k=1}^q \mathbf{Pr}([\overline{\mathcal{V}}_k]_{t+1} | [\mathcal{V}_k]_t).$$

By means of words, solving (4) brings the soft clustering whose components have *optimal* stability, and whose associated distributions are given by \tilde{Y} . As a consequence, we easily obtain that $\tilde{\mathbf{y}}_1 = \pi$, the stationary distribution. This is natural, as this is the observed distribution of types, *i.e.* the one that best explains the data.

4 Conclusion

In this paper, we have provided a novel way to interpret the results of spectral decomposition, in terms of soft clustering. This probabilistic interpretation allows to avoid the complexity gap that follows from traditional hard spectral clustering. This brings a natural approach to process a Markov chain, and make a soft clustering out of its state space. We bring an extensive comparison of hard and soft spectral clustering, along with some extended results on conventional spectral clustering.

In future works, we plan to drill down our soft spectral clustering results, to converge towards a complete probabilistic interpretation of spectral decomposition. Another target of our future research is using this method on other matrices computed from a text, like the matrix of distributional similarity (measuring “paradigmatic” syntactic distance instead of “syntagmatic” syntactic distance, cf. Introduction), with the aim of clustering syntactic and semantic categories in loosely-described languages.

REFERENCES

- [1] F. R. Bach and M. I. Jordan, ‘Learning spectral clustering’, in *NIPS* 16*, eds., S. Thrun, L. Saul, and B. Schoelkopf. MIT Press, (2003).
- [2] M. Belkin and J. Goldsmith, ‘Using eigenvectors of the bigram graph to infer morpheme identity’, in *Proc. of the ACL’02 Workshop on Morphological and Phonological Learning*. (2002).
- [3] M. Belkin and P. Niyogi, ‘Laplacian eigenmaps and spectral techniques for embedding and clustering’, in *NIPS* 14*, eds., T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, (2001).
- [4] M. Meilă and J. Shi, ‘Learning segmentation by random walks’, in *NIPS* 14*, eds., T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, (2001).
- [5] B. Mohar, ‘Some applications of Laplace eigenvalues of graphs’, in *Graph symmetry: algebraic methods and applications*, eds., G. Hahn and G. Sabidussi, 225–275, NATO ASI Series, (1997).
- [6] A. Y. Ng, M. I. Jordan, and Y. Weiss, ‘On spectral clustering: Analysis and an algorithm’, in *NIPS* 14*, eds., T. G. Dietterich, S. Becker, and Z. Ghahramani, Cambridge, MA, (2001). MIT Press.
- [7] J. Shi and J. Malik, ‘Normalized cuts and image segmentation’, *IEEE TPAMI*, **22**, 888–905, (2000).

Tools for Text Mining over Biomedical Literature

Fabio Rinaldi and Gerold Schneider and Kaarel Kaljurand and Michael Hess¹

Abstract.

This poster describes OntoGene: an environment, based on a deep-linguistic parser, aimed at supporting the process of Text Mining from Biomedical Scientific Literature. We will illustrate in particular the Relation Extraction component and the development support facilities.

1 Introduction

The Biomedical domain is witnessing a rapid growth of the amount of published scientific results, which makes it increasingly difficult, even for the domain experts, to rapidly identify the information that they need. PubMed is the best-known resource: it offers a keyword search over the entire collection of published articles (currently more than 14 million). Alternatively the experts can make use of ‘curated’ resources, which attempt to store in a knowledge-rich format the most significant results presented in the literature. Well-known examples are the Gene Ontology and the Kyoto Encyclopedia of Genes and Genomes. The creation of such resources is a very labour intensive process, therefore a (partial) automation of this activity is highly desirable. The first step is the identification of all biological relevant entities (genes, proteins, diseases, etc.). This task has been addressed quite extensively by the research community, as witnessed by events such as BioCreAtIVe. The task is made particularly difficult by the high ambiguity of the entity names in this domain: not only is there a high degree of polysemy and synonymy, also very common words can be used as names of entities.

As our aim is to show how a deep-linguistic approach can be used in a Text Mining application, offering high-precision relation extraction (while at the same time retaining a high recall), we have sidestepped the problem of entity extraction, either by using manually annotated resources, or by adopting external tools for entity detection. Our attention has been instead focused on the next step, which is the detection of the possible interactions among the entities mentioned in a document. Our results have been validated on the manually annotated GENIA corpus [4]. Separate experiments have been performed using automatic annotation tools [3].

Section 2 describes in detail our approach to relation mining, while section 3 briefly describes the two corpora on which we have performed our experiments and presents evaluation results. For a survey of related work, the reader is invited to consult [4].

2 Corpus Analysis and Relation Mining

The analysis of the documents is based on a broad-coverage probabilistic Dependency Parser [5]. The parser is wrapped into a pipeline which uses a number of other NLP tools, as described below. The output of the pipeline provides the basis for our relation mining and knowledge discovery approach.

¹ Institute of Computational Linguistics, IfI, University of Zurich,
{rinaldi, gschnied, kalju, hess}@ifi.unizh.ch

The pipeline performs a sequence of processing tasks, described below.²

1. Replace given terms with their heads
2. Sentence splitting by MXTERMINATOR
3. Tokenization by the Penn treebank tokenizer
4. Part-of-speech tagging by MXPOST
5. Lemmatization by morpha
6. Terminology detection by matching the token stream against existing term lists from biomedical ontologies
7. Noun and verb group chunking by LTCHUNK
8. Detection of chunk heads by a simple pattern matching over the part-of-speech tags of the tokens
9. Dependency parsing

When the pipeline finishes, each input sentence has been annotated with additional information: sentences are tokenized and their borders are detected; each sentence and each token has been assigned an ID; each token is lemmatized; tokens which belong to terms are grouped; each term has a normal-form and a semantic type; tokens and terms are then grouped into chunks; each chunk has a type (NP or VP) and a head token; each sentence is described as a syntactic dependency structure; each dependency occurs between two tokens and has a type.

The parser expresses distinctions that are especially important for a predicate-argument based deep syntactic representation, which includes PP-attachment, most long distance dependencies, relative clause anaphora, participles, gerunds, and the argument/adjunct distinction. The parser is very robust and has been applied to parsing large amounts of text data, including the 100 Million word British National Corpus.

Our approach to relation mining is based on 3 levels of rules. On the first level, we exploit simple *syntactic patterns* detected in the data. On the second level we combine various patterns into a single *semantic rule*, which does away with many possible syntactic variants (e.g. active, passive, nominalizations). On the third level, we combine semantic rules with lexical and ontological constraints to obtain very specialized queries that can detect a given domain-specific relation, as specified by the user.

Syntactic rules capture general linguistic phenomena and as such are highly reusable across different domains. Simpler rules can easily be combined into more complex rules - thus making the system more modular. An example of a syntactic rule for the passive case is shown below:³

```
synRel(simple_passive, [H,B,A],  
       [ dep(subj,H,B), dep(pobj,H,A), dep(prep,A,By),  
         pos(H,'VBN'), lemma(By,['by','through','via']) ] ).
```

The next step is then to combine different syntactic patterns to yield a *semantic rule*. For example, once we have defined a syntactic

² Some of these steps (e.g. tagging, terminology detection) are not necessary - and are automatically skipped - if the relevant information is already provided in the Corpus.

³ A prolog-based syntax is adopted.

pattern for the active case (e.g. *A regulates B*), another pattern for the passive case (*B is regulated by A*), and a pattern for the nominalized case (*The regulation of B by A*), we can define a semantic relation which combines these possible syntactic patterns into a single relation SemRel(H, A, B). Once such a relation is defined, it can be used in query, like the following:

```
applyRel(SemRel([regulate,A,B]))
```

which returns all the sentences containing a *regulate* relation, where A and B are instantiated respectively by the agent and the target of the relation, irrespective of their syntactic realization.⁴

The query mechanism has been further extended to allow the user to match semantic types of the arguments, if available in the data. For example in the GENIA corpus, there is a semantic type called G#Protein, and we can use the following query to retrieve all ‘activate’ relations where the agent is an entity belonging to this type:

```
applyRel(SemRel([activate,type:'G#Protein',B]))
```

If a domain Ontology is available, our system can make use of it in the query process, extending the interpretation of the type restriction to refer not only to the objects that directly match the given type, but also to those that are subsumed by it.

As the set of patterns and rules is gradually enriched, so are the possible lexico-syntactic variants that can be captured. An example of a more advanced rule is the one which captures “*A triggers the H of B*”, where H represent a nominalized verb (*activation, regulation, etc.*). Similar complex rules have designed, e.g. for “*under the control of*”, “*involved in*”, “*be able to*” etc. Because such rules are built on top of the lower-level rules, they automatically capture the known syntactic variants, such as “*The H of B is triggered by A*”. We refer to relations defined at this level as *domain relations* as they rely on lexical constraints which are typical of a given domain. The user query can happen at each one of the 3 levels. So it is possible to test individual syntactic rules, semantic rules, and domain rules.

3 Experiments and Evaluation

The tools described in this paper have been applied to extract semantic relations from 2 distinct corpora. GENIA [2] is a corpus of 2000 Medline abstracts which have been *manually* annotated (by domain experts) for various biological entities according to the GENIA Ontology. The ATCR Corpus (Arabidopsis Thaliana Circadian Rhythms) is a corpus of 147 Medline abstracts (up to year 2004), extracted using the keywords: “Arabidopsis Thaliana” and “Circadian Rhythms”. It has been *automatically* annotated using the “Biolab Experiment Assistant (BEA)”TM.⁵

Based on the interaction with a domain expert, we have identified a set of relations that are of particular interest in this domain. Some examples of relevant relations are: *activate, bind, block, regulate, control, express*. We have designed the extraction algorithm to maximally expand the arguments of the predicate, following all their dependencies. Each argument is then assigned a type based on its head. We then removed all records where a type had not been assigned to either agent or target. This remaining set was inspected by a domain expert. We asked the domain experts to evaluate each relation and each argument of the relation, and mark them according to its correctness and its biological significance.

⁴ We refer to the argument A as the agent, to B as the target of the relation, adopting the terminology used in [1]. The first argument (H) defines the type of the relation (e.g. “*regulate*”).

⁵ Due to lack of space only results on the former are reported here, results on the latter can be found in [3]. For further information on BEA please see <http://www.biovista.com/>.

	agent				target			
	Y	A	P	N	Y	A	P	N
activate	72	64	5	8	77	54	8	10
bind	36	18	1	8	39	18	1	5
block	3	0	0	0	1	1	0	1
TOTAL	111	82	6	16	117	73	9	16
	52%	38%	3%	7%	55%	34%	4%	7%
	correct	90%	incorrect	10%	correct	89%	incorrect	11%

Table 1. Analysis of precision

Corpus	Relation	Observed or Estimated Recall
ATCR (observed)	control regulate	60% 60%
GENIA (estimated)	control regulate	38 - 41% 50 - 65%

Table 2. Estimate of recall. Extrapolated percentages are in **boldface**

The evaluation on the GENIA corpus results in the values shown in table 1. The results show that we obtain precision of at least 50% exactly correct [Y] for both agent and target. Including the cases where only a part of the relevant information or too much information is highlighted [A] leads to a precision of about 90%. The expert browsing these results, easily and quickly corrects these over- or underexpansions, scanning the context few words to the left or right is sufficient. Unresolved pronouns [P] need a reader to deal with a substantially larger context. Our system does not yet have pronoun resolution. We have therefore decided to report these cases as incorrect.

In the absence of a gold standard, only approximative recall values can be reported. In [3] we report a value of 40% for a measure that we call “worst-case recall”, which basically implies that our actual recall is at least as good as this value. On a smaller subset of the corpus we actually measured a recall value of 60%. Using the recall obtained on the development set and the (measurable) coverage on the test set, we can estimate the value of recall on the latter. The results can be seen in table 2.

Additionally, the OntoGene Text Mining environment provides facilities for debugging and visualization. For example, each result bears the name of the rules that generated it. This allows immediate detection of problems and their quick correction. We also provide a “visual diff” facility that shows in the same graphical format the matches that have been acquired or lost as a consequence of the addition of a new pattern or rule. For further information and examples please visit <http://www.ontogene.org/>.

REFERENCES

- [1] J. Cussens and C. Nédellec, eds. *Proceedings of the workshop "Learning Language in Logic (LLL05)"*, 2005.
- [2] J.D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, ‘GENIA Corpus - a Semantically Annotated Corpus for Bio-Textmining’, *Bioinformatics*, **19**(1), 180–182, (2003).
- [3] Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, Christos Andronis, Ourania Konstanti, and Andreas Persicid, ‘Mining of Functional Relations between Genes and Proteins over Biomedical Scientific Literature using a Deep-Linguistic Approach’, *Journal of Artificial Intelligence in Medicine (accepted for publication)*, (2006).
- [4] Fabio Rinaldi, Gerold Schneider, Kaarel Kaljurand, Michael Hess, and Martin Romacker, ‘An Environment for Relation Mining over Richly Annotated Corpora: the case of GENIA’, in *Proceedings of SMBM 2006, Second International Symposium on Semantic Mining in Biomedicine*, pp. 68–75, Jena, Germany, (April 9-12 2006).
- [5] Gerold Schneider, Fabio Rinaldi, and James Dowdall, ‘Fast, Deep-Linguistic Statistical Minimalist Dependency Parsing’, in *COLING-2004 workshop on Recent Advances in Dependency Grammars, August 2004, Geneva, Switzerland*, (2004).

SUMMaR: Combining Linguistics and Statistics for Text Summarization

Manfred Stede and Heike Bieler and Stefanie Dipper and Arthit Suriyawongkul¹

Abstract. We describe a text summarization system that moves beyond standard approaches by using a hybrid approach of linguistic and statistical analysis and by employing text-sort-specific knowledge of document structure and phrases indicating importance. The system is highly modular and entirely XML-based so that different components can be combined easily.

1 INTRODUCTION

Most text summarization systems nowadays take the approach of *sentence extraction*: Following the determination of most relevant terms in the document, sentences are selected that contain such relevant terms, and the sequence of these sentences is deemed the summary. Term relevance is traditionally measured by relative frequencies in different corpora; various extensions of this basic method are being used, such as the position of sentences in documents (e.g., prefer sentences at the beginning of the document or at the end).

In contrast to generic summarization systems that emphasize robustness (produce *some* summary for *any* text), our project trades breadth against quality and argues that good summarization should be *text-sort-specific*. The idea is that the notion of ‘importance’ is relative to the sort of text; for example, in a news story, the most important information is the event stated right at the beginning, whereas for an op-ed piece, the most important information is the topic plus the opinion the author conveys. Our approach is to use a declarative representation of text-sort knowledge that supplements the standard extraction technique in creating the summary. With SUMMaR², we created a highly modular, XML-based architecture that allows for plugging in various analysis tools and combining their results flexibly. A GUI displays interim analysis results as well as the final summary, thus facilitating further fine-tuning. SUMMaR is an implemented prototype with some components still under improvement. Among the text sorts we are working with (news, commentary, scientific papers, reviews), reviews are currently most prominent; in particular, here we use *movie reviews* to illustrate our approach.

2 OVERVIEW OF SYSTEM ARCHITECTURE

For text analysis in any application, flexible module combination is an important desire. We achieve this by consistently using *standoff annotation*: For each information layer, a single XML representation is created, which points either to the tokenized source document, or to another layer (e.g., PoS layer points to source text, whereas syntax-chunk layer points to PoS layer). Details of the XML format can

be found in [Dipper 2005]. The analysis modules can then inspect layers individually or create combinations of them by merging them to a standard inline XML representation; for this purpose, we use the approach of [Witt et al. 2005].

SUMMaR processes documents in plain text, XML, and (to some extent) HTML. The first step is to map these input formats to a common XML format representing the basic layout of the text, i.e., the structure of headlines and paragraphs (see Section 3). Then, tokenization identifies sentence and token boundaries. This is input to a syntactic parser followed by co-reference analysis; these steps are taken to identify cohesion problems in the resulting text extracts, which will later be resolved by partial re-generation.

For developing the system, it has proven useful to employ a GUI that displays the source text and, on demand, the various annotations as produced by analysis modules. This could to some extent be achieved by configuring a workbench such as GATE (<http://gate.ac.uk>), but we found it more comfortable to write our own web-based GUI geared specifically to the needs of summarization, so that, e.g., the most relevant sentences of the text can be highlighted dynamically when changing relevance thresholds, etc.

We claim that for quality summarization, the system needs to know the *text sort* of the document under consideration: It makes a difference for the summarization process whether, for example, a news report or a company white paper or a review article is being handled. The key is to encode these differences in a declarative knowledge source so that the processing modules can in fact stay the same. One piece of text-sort-specific information is the *target template* for the summary: For movie reviews, it should contain the source of the review, movie title, the assigned overall rating (if any), indication of the story, and indication of more specific judgements of the reviewer. Such a task-oriented summarization is in effect a mixture of traditional information extraction (“find the title; find the overall rating”) and traditional summarization (“condense the description of the story, and the evaluative portions, if any”).

Our module for *content structure* identification employs text-sort knowledge in order to assign labels to paragraphs similar to the *zones* in [Teufel and Moens 1997]. When zones have been identified, their contents are either directly transferred to the target template (for movie reviews: title and overall rating), or they are subject to *sentence relevance* calculation, as described in Section 4.

3 DOCUMENT STRUCTURE KNOWLEDGE

The basis for our text analysis modules is an XML format that marks paragraphs, headings, and emphasized text portions (e.g. italics, bold face). It represents the logical document structure in a sense similar to a Latex source, albeit much simpler. Considerably less simple is the procedure to derive this logical structure from different types of

¹ University of Potsdam, Germany, email: stede|bieler|dipper|art@ling.uni-potsdam.de

² Project funded by Bundesministerium für Bildung und Forschung, grant 03WKH22. The authors are responsible for the content of this paper.



Figure 1. Excerpt from movie review webpage

source documents. We process different kinds of XML formats (from our partner projects), plain text, and to some extent HTML. The current implementation of our logical structure extractor is written in Python, with optional help from an XSL Transformation engine. For the case of plain text documents, the only layout information is the use of spacing, vertically and horizontally, and some creative typography like using asterisks around characters for **emphasis**. We built a set of heuristic rules that identify paragraph breaks on the basis of average line length and presence of single and double line breaks in the document; similarly, headings are identified on the basis of length, cue words (like numbering), and surrounding line breaks.

Having derived the logical structure, we enrich it with information on content zones. The inventory of zones has been determined by a corpus study. For movie reviews, it includes labels such as Title, Rating, DescStory, CommentOnStory, CommentOnActors, Credits, LegalNotice. Our corpus includes 50 reviews from 10 different sources, and so the documents differ widely, both in length and in structure. There are regularities, however: Title is at the beginning; legal notice, if any, at the end; overall rating either towards beginning or towards end; story is usually told continuously (no intervening non-story paragraphs), and so on. We have formalised the regularities not as a document grammar, but as a set of local rules, and the process of assigning each paragraph of the logical structure a content-label is one of constraint satisfaction with optimization. These steps of inferring logical and content structure are described in detail in [Stede and Suriyawongkul, to appear].

4 SUMMARY PRODUCTION

As mentioned earlier, the first items of the summary template (title and rating) are determined by simple information extraction rules from the content structure representation. For the indicative summary of the story, only those paragraphs that have received a 'Story' label are submitted to our sentence relevance calculation, which is based on computing term weights with the $tf * idf$ method, where the relation of the frequency of the term in the specific document and the number of documents in which this terms occur, plays a role. As to the notion of 'term', we are experimenting with wordforms (easy to

detect, but with inflection problem), stems as determined by a Porter-stemmer (resolves inflection problems, but overgenerates), n grams (all combinations of n adjacent characters in the text; a very robust method that can cope with spelling errors), and lemmas (for applications where a lemmatizer for the language is available). The $tf * idf$ measure counts how often a term occurs in the document (term frequency tf), and in how many documents of our corpus for the specific text sort (document frequency df). Terms with high term frequency and a low document frequency are the most indicative terms, because they reflect the specific topic (in this case, of the movie).

For a text like that of Figure 1, the method would find terms such as *East German's*, *Lenin*, *amnesia* – but it would not find terms such as *popular*, *seamlessly*, *satisfying*. The latter are typical for any opinion-conveying text and thus will be frequent in a corpus of reviews of any kind, including movies. In order to determine the reviewer's opinion (which should be reflected in a good summary), we look for sentences including such evaluative expressions (currently from a hand-collected list, but this is being replaced by a trained statistical classifier), and use these sentences as the basis for the final entry of the summary template, 'Evaluation'. This is an important difference to other approaches to sentence extraction: text-sort-specific terms (here: evaluative ones) are used to determine *important* sentences, whereas the $tf * idf$ measure is used to determine sentences that are *indicative* of the content. The list of evaluative phrases is thus part of the text-sort knowledge (albeit in this case shared between movie reviews and all kinds of other product reviews), which collectively is represented as a distinct XML document. It contains the information on necessary and possible zones for the text sort, and terms acting as indicators for zones, such as the above-mentioned evaluative terms, or a number of asterisks for the 'Rating' Zone (e.g., in Figure 1: "****1/2 (our of four)". In the end, the filled output template for the text in Figure 1 would look like this:

Source: Film Freak Central, Feb 22, 2006
Title: Good Bye, Lenin (2003)
Rating: ***1/2 (out of four)
Story: <extract from story description>
Evaluation: <extract from evaluative portions>

5 CONCLUSION

Document analysis is to a large extent a matter of statistical relevance calculations, but it should also be driven by information on document structure. We have illustrated this for the case of text summarization: Given loosely-structured documents consisting of a fairly predictable set of content zones (but not in a fixed order; otherwise it is a highly-structured document), we propose to first identify this content structure as a useful step of preprocessing. For summarization, this helps to make sure that portions of all relevant zones are actually part of the result.

REFERENCES

- [Dipper 2005] S. Dipper. "XML-based Stand-off Representation and Exploitation of Multi-Level Linguistic Annotation." In: Proc. of Berliner XML Tage 2005, pp. 39-50, Berlin, Germany.
- [Stede and Suriyawongkul, to appear] M. Stede, A. Suriyawongkul. "Identifying Logical Structure and Content Structure in Loosely-Structured Documents." To Appear.
- [Teufel and Moens 1997] S. Teufel, M. Moens. "Sentence extraction as a classification task." In: Mani and Maybury, eds.: *Advances in Automatic Text Summarization*, MIT Press, 1997.
- [Witt et al. 2005] A. Witt, H. Lüngen, F. Sasaki, D. Göcke. "Unification of XML Documents with Concurrent Markup." *Literary and Linguistic Computing* 20(1), 103-116, 2005.

Phonetic Spelling and Heuristic Search

Benno Stein¹ and Daniel Curatolo²

Abstract. We introduce a new approach to spellchecking for languages with extreme phonetic irregularities. The spelling for such languages can be significantly improved if knowledge about pronunciation and sound becomes the central part of the spelling algorithm. However, given a weak phoneme-grapheme-correspondence the standard spelling algorithms, which are rule-based or edit-distance-based, are severely limited in their phonetic capabilities.

A production approach to spelling can overcome the limitations—but suffers from its search space size. We describe in this paper the main building blocks to tackle this problem with heuristic search. Our ideas have been operationalized in the SMARTSPELL algorithm, with impressive results related to spelling correction and runtime.

1 INTRODUCTION AND PROBLEM SETTING

Orthography is the study or practice of correct spelling or writing; spelling is the choice of which letters or symbols to combine to represent a word. A (single word) spelling algorithm takes a dictionary D and a possibly misspelled word w as input and returns the most similar words w_1, \dots, w_k from D with respect to w . The “quality” or “power” of a spelling algorithm depends on the operationalized similarity measure, implicating an inevitable tradeoff between runtime performance and spelling quality.

The difficulty of a spelling problem depends on both the type of the spelling error to be detected and the underlying language. Table 1 illustrates common types of spelling errors with respect to the single word spelling problem.

Spelling error type	Example
Permutations or dropped letters	hpantom → phantom
Misremembering spelling details	recieve → receive, remembering believe
Trying to spell out pronunciation	tuleboks → toolbox

Table 1. A hierarchy of spelling errors with increasing problem complexity.

A language’s key impact to spelling is governed by its *phonemicity*, which describes the extent to which spelling is a guide to pronunciation: A word is called phonemic if its spelling corresponds to its pronunciation. However, Trost pointed out that written language is rarely a true phonemic description [10]. Table 2, taken from an IEA survey [3], arranges languages with respect to their degree of phonemicity.

One of the many examples for the high irregularity of the English language is the word “school” where among others the following writings produce the same sound: skool, scool, scule, skule. The sources of irregularity are positional spelling, i. e., the sound of letters varies according to the position in a word, and polyvalence, i. e., letters can produce different sounds.

¹ Faculty of Media / Media Systems. Bauhaus University Weimar, Germany.
 benno.stein@medien.uni-weimar.de

² Art Systems Software GmbH. Paderborn, Germany.

Highly regular <		Spelling system	-> Irregular	
Finnish	Spanish, Portuguese, Italian, Hungarian, Slovenian	German, Dutch, Greek Swedish, Norwegian, Icelandic	Danish, French	English

Table 2. Languages ordered with respect to their degree of phonemicity [3].

Considering the sound of a written word within a spelling process is by far more complex than simply computing edit distances, since extra combinatorics is introduced at two places: (i) Which letters form a group that produces a single sound? (ii) Which sound is produced? This paper shows how heuristic search can be used to significantly speed up the similarity analysis in the huge search space. The operationalization of the presented ideas led to an efficient algorithm, SMARTSPELL, which is used in real-world applications.

The paper is organized as follows. The remainder of this section classifies possible solutions to the spelling problem, and Section 2 outlines the main ideas of the heuristic search behind SMARTSPELL.

1.1 Handling Misspelling

To classify existing work on spelling correction for single words we developed the taxonomy shown in Figure 1.

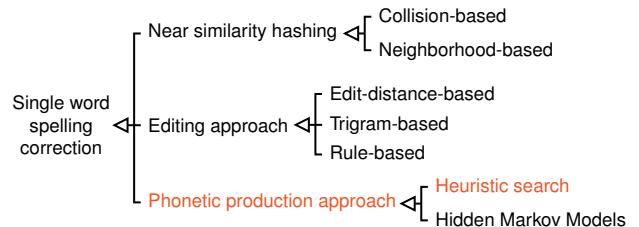


Figure 1. Classification of spelling correction methods.

At its top-level the taxonomy distinguishes three main approaches: (i) approaches that rely on the computation of hash keys for the words in D , (ii) editing approaches, which transform a word $w \notin D$ into a word from D , and (iii) phonetic production approaches, which construct a letter sequence that produces the observed sound.

The hashing approaches further subdivide into methods that exploit hash collisions (e. g. Soundex [6]) or evaluate neighborhoods in the list of the sorted hash keys (cf. SpeedCop [9] or [7]). Today, many approaches to handle misspelling are based on edit distances: The candidates for the correct spelling of some word w minimize the number of editing operations that transform w into some word of a dictionary D . The Levenshtein distance is a well-known measure to operationalize this idea; it associates each substitution, deletion, or insertion of a letter with certain cost [4]. This idea can be applied to trigrams [1] or become refined by phonetic transformation rules (cf. Aspell [2] or Correct [5]).

2 PHONETIC SPELLING VIA SEARCH

We understand phonetic spelling as the generation of the most probable letter sequence with respect to phonological interpretation. Such a production approach may be realized with Hidden Markov Models or with heuristic search [8]; it is much more ambitious than an editing approach with respect to the explored search space. Note in this connection that editing approaches are too limited to be used as a guide to phonetic similarity. Examples:

- The Levenshtein distance between the words “eaecutioib / execution” and “aksekjushen / execution” is 2 and 8 respectively. Observe that the phonetic similarity in the second case is significantly higher than in the first case.
- The phonetically motivated edit rule “k → c” works fine to correct the misspelling “kat / cat”, but it fails in the case “cule / cool”.

The following three concepts—here applied to phonetic spelling—together form the elements of heuristic search:

1. *Search Space*. Defines all segmentations of all words in the dictionary D , along with operators to move between them.
2. *Evaluation Function ϕ* . Quantifies the quality of a segmentation of an unknown word with respect to the entire search space.
3. *Control Strategy*. Guides the exploration of the search space.

Search Space Since a hyphenation-based segmentation may not be optimum with respect to phonetics, the search space of SMARTSPEL contains for each word $w \in D$ the set Π_w of all possible segmentations, where each segment $s \in \pi_w$ in a segmentation $\pi_w \in \Pi_w$ is either a letter, 2-, 3-, or 4-gram. Hence, possible segmentations π_w for $w = “execution”$ are:

e - x - e - c - u - t - i - o - n, ex - e - cut - i - on, e - x - ecu - tio - n

Altogether, “execution” has $|\Pi_w| = 773$ segmentations; a word of length n has $\sum_{i=1}^n \text{seg}(n-i)$ segmentations, with $\text{seg}(0) = 1$.

Evaluation Function ϕ $\phi(v, w, \pi_v, \pi_w)$ computes the similarity between two words, v, w , provided two segmentations, π_v, π_w ; it is based on the phonetic a-priori similarity, $\varphi(s_i, s_j)$, for two segments s_i, s_j . φ is language-dependent and defines about 2500 segment similarities for the English language, among others:

$$\begin{array}{llllll} s_i & s_j & \varphi(s_i, s_j) & ac & ec & .95 \\ & & & ac & ec & .80 \\ & & & ac & eck & .85 \end{array} \quad \begin{array}{llllll} ad & ed & .95 & af & ef & .95 \\ ad & edd & .90 & af & eff & .90 \\ ad & eg & .57 & af & es & .38 \end{array}$$

For the 24 essential sounds of RP English (received pronunciation) we developed a new concept to construct the function φ automatically: Based on so-called sound contexts, which are activated by the different consonant types like plosives, nasals, etc., raw estimates for the similarities are computed and statistically smoothed.

Let $\pi_v = s_{v,1} \dots s_{v,|\pi_v|}$ and $\pi_w = s_{w,1} \dots s_{w,|\pi_w|}$, then ϕ is defined as follows:

$$\phi(v, w, \pi_v, \pi_w) = \frac{\sum_{i=1}^{\min\{|\pi_v|, |\pi_w|\}} (|s_{v,i}| + |s_{w,i}|) \cdot \varphi(s_{v,i}, s_{w,i})}{|v| + |w|}$$

For example, the computation of ϕ for the two pairs of segmentations e-x-e-c-u-ti-o-n / a-ks-e-k-ju-sh-o-n and e-x-_e-c-u-ti-o-n / a-k-s-e-k-ju-sh-o-n yields the following values:

$$\begin{array}{llll} \varphi(s_i, s_j) & & & \\ e & a & .70 \cdot 2 & = 1.40 \\ x & ks & .80 \cdot 3 & = 2.40 \\ e & e & 1.00 \cdot 2 & = 2.00 \\ c & k & .95 \cdot 2 & = 1.90 \\ u & ju & .75 \cdot 3 & = 2.24 \\ ti & sh & .90 \cdot 4 & = 3.60 \\ o & o & 1.00 \cdot 2 & = 2.00 \\ n & n & 1.00 \cdot 2 & = 2.00 \end{array} \quad \begin{array}{llll} e & a & .70 \cdot 2 & = 1.40 \\ x & k & .30 \cdot 2 & = .60 \\ s & & 0 \cdot 2 & = 0 \\ e & e & 1.00 \cdot 2 & = 2.00 \\ c & k & .95 \cdot 2 & = 1.90 \\ u & ju & .75 \cdot 3 & = 2.24 \\ ti & sh & .90 \cdot 4 & = 3.60 \\ o & o & 1.00 \cdot 2 & = 2.00 \\ n & n & 1.00 \cdot 2 & = 3.00 \end{array}$$

$$\phi = 8.77/(9+11) \Rightarrow 88\% \quad \phi = 7.87/(9+11) \Rightarrow 79\%$$

From all segmentations, Π_v, Π_w , for two words, v, w , one is interested in those that maximize ϕ ; they are implicitly defined by ϕ^* :

$$\phi^*(v, w) = \max_{\pi_v \in \Pi_v, \pi_w \in \Pi_w} \phi(v, w, \pi_v, \pi_w)$$

ϕ fulfills the standard properties of a similarity measure; i.e., it is normalized, reflexive, and symmetric. In addition, it has a monotonic characteristic in word lengths: $|v| < |u| \Rightarrow \phi(w, uv) > \phi(w, uw)$

Control Strategy Checking the spelling of a word w regarding its phonetics means to identify a word $v \in D$ such that two segmentations $\pi_v \in \Pi_v$ and $\pi_w \in \Pi_w$ can be found that maximize ϕ . A typical value for $|D|$ is 10^6 , a typical value for $|\Pi_w|, w \in D$, is $> 10^2$. Hence, the size of the search space for an ordinary spelling query is in $O(10^{10})$. To achieve an acceptable response time SMARTSPEL operationalizes a sophisticated control strategy that combines the following elements:

- Segmentation Heuristic. Segmentations are constructed stepwise, striving for a minimum length difference of the residual strings.
- Early Pruning. Exploitation of the monotonicity of ϕ for pruning.
- Iterative Deepening. Candidates for a Depth-First Search are chosen from a pool whose pruning threshold is successively lowered.
- Nogood Construction. Segments that are unlikely to match are stored in a special nogood table for φ .
- Memorization. Strings with high ϕ -values are remembered.

Results Following examples illustrate the power of the phonetic production approach and the SMARTSPEL algorithm:

angenehing	→	engineering 92%
buysikel	→	physical 85%, bicycle 82%
dshungal	→	jungle 90%
hachock	→	hedgehog 95%
jentelman	→	gentleman 85%
kompilayshon	→	compilation 89%
refridjeraitar	→	refrigerator 79%
siantifik	→	scientific 87%
tradisionell	→	traditional 93%
tshylt	→	child 97%
tulebogs	→	toolbox 93%

The function φ was constructed for English, German, and Spanish; moreover, SMARTSPEL has proven its usability in several real-word applications. On request we will provide Web-based access.

REFERENCES

- [1] R. Angell, G. Freund, and P. Willett, ‘Automatic Spelling Correction Using a Trigram Similarity Measure’, *Information Processing and Management*, **19**(4), (1983).
- [2] K. Atkinson. Aspell. aspell.sourceforge.net/, 2004.
- [3] W. B. Elley, ‘How in the World do Students Read?’, Technical report, The Hague, International Association for the Evaluation of Educational Achievement (IEA), (1992).
- [4] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, 1997.
- [5] B. Kessler. A spelling corrector incorporating knowledge of English orthography and pronunciation. www.artsci.wustl.edu/~bkessler/correct, 1998.
- [6] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, 1973.
- [7] K. Kukich, ‘Spelling correction for the Telecommunications Network for the Deaf’, *Com. of the ACM*, **35**(5), (1992).
- [8] J. Pearl, *Heuristics*, Addison-Wesley, Massachusetts, 1984.
- [9] J. Pollock and A. Zamora, ‘Automatic spelling correction in scientific and scholarly text’, *Com. of the ACM*, **27**(4), (1984).
- [10] Harald Trost. Computational Morphology. www.ai.univie.ac.at/~harald/handbook.html, 2001.

8. PAIS

This page intentionally left blank

CBR-TM: A new Case-Based Reasoning System for Help-Desk Environments

Juan Ángel García-Pardo¹ and Stella Heras Barberá¹ and Rafael Ramos-Garijo¹
 and Alberto Palomares² and Vicente Julián¹ and Miguel Rebollo¹ and Vicent Botti¹

Abstract. In this paper, a new CBR system for help-desk environments is presented. This CBR system provides intelligent customer support for multiple domains. It is also portable and flexible. The system is implemented as a module of a complete help-desk application to make it as independent as possible of any change in the help-desk system. Each phase of the reasoning cycle is also separated as an independent module, making the CBR system easy to update. The system has been tested in a real call center managed by the Spanish company TISSAT S.A.

1 Introduction

Customer support has become a profitable business. A quick and accurate response to customer problems ensures their satisfaction and a good reputation for the company. Companies usually attend to customers via a call center, where a number of operators manage the queries using a help-desk application. To provide high-quality customer support the operators must be trained. Moreover, the knowledge of experienced operators should be saved in order to maintain it over the course of time or if the operators leave the company. Reusing successful past solutions may be a suitable method to improve the performance of a customer support system.

A Case-Based Reasoning system (CBR) tries to solve a new problem (case) by means of reusing the solution of a past similar case. This solution is previously stored in a case memory (case-base) and can either be retrieved and directly applied to the current problem, or revised and adapted to fit the problem. The successful application of CBR systems in help-desks for managing call centers has proved the suitability of these systems since the 90s [2][5]. There are also many companies that sell software for applying CBR to help-desks. We were asked to implement a new CBR system as generic as possible to provide intelligent customer support to a help-desk environment without using any copyrighted software or any CBR tool that is only available for research purposes. Even though it has been tested in a specific help-desk that provides technical support for computer systems, one of our main objectives was to develop a modular CBR system which can be easily adapted and applied to any domain related to customer support.

¹ Information Systems and Computing Department, Universidad Politécnica de Valencia, Spain, email: sheras@dsic.upv.es

² TISSAT S.A. <http://www.tissat.es>

2 The CBR-TM module

The Spanish company TISSAT S.A [3] provides customer support for public administration organisms and private companies. In order to improve the service, TISSAT has developed a help-desk toolkit called I2TM (Intelligent and Integrated Ticketing Manager, version 6). Each request received by TISSAT is called *ticket* and is tracked by means of the I2TM application. This application works with either queries related to computer errors (network systems failures and personal computer problems) or with requests of citizens about public services (such as the international emergency phone number 112 of the Valencia region, which covers the emergencies of over four and a half million people). Therefore, I2TM must be able to rapidly solve requests from very diverse domains.

The CBR system that we have developed, called CBR-TM (Case Based Reasoning for Ticketing Management) acts as an intelligent module for the I2TM system, helping it with the problem-solving process. Implementing CBR-TM as a module allows both systems to be changed and updated independently. We have integrated CBR-TM into I2TM using web-services. This approach was chosen because of its flexibility and distributed access possibilities. The webservices work as a bridge between I2TM and CBR-TM. The overview of the entire system architecture is shown in Figure 1.

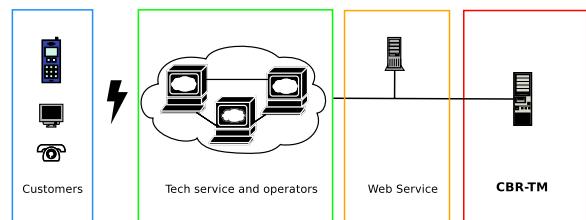


Figure 1. I2TMv6 architecture overview

The CBR methodology for problem solving is typically divided into the phases of the CBR cycle [1]: *Retrieve*, *Reuse*, *Revise* and *Retain*. The design decisions adopted for each phase in CBR-TM were principally influenced by the customer support domain and our goal to provide flexibility. In CBR-TM, a case is the prototyped representation of a set of tickets with the same features and the same successfully applied solutions. Each case has a set of attribute-value pairs (variables of any value type) that describe the characteristics

of the problem. The indexing schema hierarchically organises the cases in the computer memory to facilitate the retrieval. The indexing process in our system consists of establishing a set of categories for each ticket that classifies it as belonging to a certain type of problem, which is currently being done manually. These categories are also stored in each case as attributes.

The main goal of the retrieval phase is to obtain the set of stored cases that are similar to the new one. Due to the multiple domains of the requests that the I2TM system receives, the CBR-TM module must be able to work with heterogeneous tickets and must also be able to compute the similarity among them. Moreover, the ticket attributes may have missing values. It would be desirable to have a modular system into which we could insert different types of retrieval algorithms, without the inconvenience of having to redesign the entire system if a new representation space is considered. Thus, we have adapted and tested several known distance measures to work with heterogeneous data. The most similar case or cases are selected by means of a *k-nearest neighbour* algorithm using these distance measures to compute the similarity among cases. Currently, we have implemented two Euclidean-based similarity measures (*Normalized* and *Classic*) and a similarity measure based on the ratio model proposed by *Tversky* [4].

Once the most similar case is selected, its solution is proposed to solve the new problem that has been reported to the system, in a simple reuse phase. At this time, the solution is copied and applied to the new problem without any changes, but it is also configurable and, if necessary, any adaption process can be introduced. When a solution has been provided, the customer must indicate to the system whether that solution has really solved the problem. By means of this revision phase, the system can learn the degree of suitability of the responses that CBR-TM provides. Finally, if there is not a case that is similar enough to the new ticket, the system stores this ticket and its solution in the case-base during the retention phase.

3 Discussion

In order to evaluate the CBR-TM module, we used a synthetic database of tickets that came from computer errors. Note that a ticket of this database is not equivalent to a case, since a case is the prototyped representation of a set of tickets with the same features and the same solution that has been applied satisfactorily in the past.

We used the ticket database to test the CBR-TM performance. This performance may be influenced by the number of tickets processed by the system or the number of customers sending requests simultaneously. The tests were performed using a *cross-partition* technique separating the ticket database into two databases for training (loading the case-base) and testing the system. We repeated the tests for each similarity measure implemented in order to analyse its behaviour in the computer error domain.

The top graph of Figure 2 shows that as the number of tickets in the case-base of the CBR-TM system increases, the mean error in the answers provided by the system decreases. This fact demonstrates that the system knowledge goes up as the amount of processed data increases and, thus, CBR-TM is learning effectively. The bottom graph of Figure 2 shows the

response time when the number of customers making simultaneous requests increases. As expected, this time increases in proportion to the number of simultaneous customers. In addition, this test shows that CBR-TM is able to answer the requests quickly even when there are a considerable number of simultaneous requests. The results of the tests also show that all the similarity measures behave in a very similar way and any of them might be suitable for this domain.

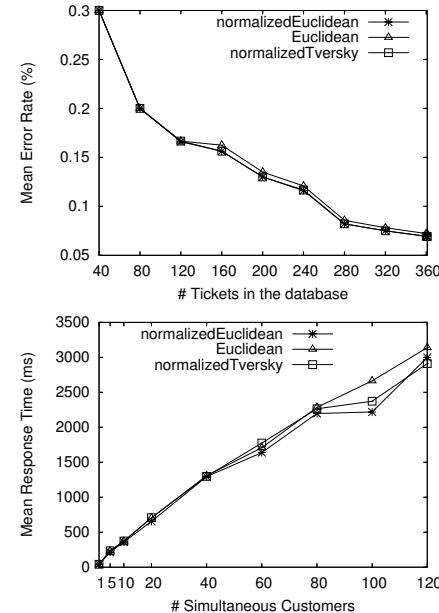


Figure 2. Top: Influence of the database size on the CBR-TM system performance; Bottom: Influence of the number of simultaneous customers on the CBR-TM system performance

Although the CBR-TM system has been tested in a help-desk whose purpose is to solve computer errors, TISSAT plans to use it in different domains in the near future. Moreover, since this system has been recently implemented, more intensive research to improve the algorithms and the techniques applied will be performed. One of our main interests is to change the manual categorisation for an automatic one. This would prevent CBR-TM from making mistakes due to human error.

ACKNOWLEDGEMENTS This project was supported by the Spanish Government under grant PROFIT FIT-34001-2004.

References

- [1] A. Aamodt and E. Plaza, ‘Case-based reasoning: foundational issues, methodological variations and system approaches’, *AI Communications*, **7**, no. 1, 39–59, (1994).
- [2] Thomas R. Roth-Berghofer, ‘Learning from HOMER, a case-based help-desk support system’, in *Advances in Learning Software Organizations*, eds., Grigori Melnik and Harald Holz, pp. 88–97. Springer-Verlag, (2004).
- [3] Tissat S.A. www.tissat.es, 2006.
- [4] A. Tversky, ‘Features of similarity’, *Psychological Review*, **84**, no.4, 327–352, (1997).
- [5] I. Watson, *Applying Case-Based Reasoning. Techniques for Enterprise Systems*, Morgan Kaufmann Publishers, Inc., California, 1997.

Verification of Medical Guidelines using Task Execution with Background Knowledge

Arjen Hommersom and Perry Groot and Peter Lucas¹
Michael Balser and Jonathan Schmitt²

Abstract. The use of a medical guideline can be seen as the execution of computational tasks, sequentially or in parallel, in the face of patient data. It has been shown that many of such guidelines can be represented as a ‘network of tasks’, i.e., as a number of steps that have a specific function or goal. To investigate the quality of such guidelines we propose a formalization of criteria for good practice medicine a guideline should comply to. We use this theory in conjunction with medical background knowledge to verify the quality of a guideline dealing with diabetes mellitus type 2 using the interactive theorem prover KIV. Verification using task execution and background knowledge is a novel approach to quality checking of medical guidelines.

1 INTRODUCTION

The trend of the last decades has been to base clinical decision making more and more on sound scientific evidence, i.e., *evidence-based medicine*, which has led medical specialists to develop medical guidelines, i.e., structured documents providing detailed steps to be taken by health-care professionals in managing the disease in a patient, for promoting standards of medical care. For their employment, computer-oriented languages are being developed. Examples are PROforma [2] and Asbru [6], which model complex clinical processes as a ‘network of tasks’, i.e., a number of steps that have a specific function or goal [2]. However, guidelines should not be considered *static* objects as new scientific knowledge becomes known on a continuous basis. Rapidly changing evidence makes it difficult to adjust guidelines in such a way as to keep them up to date.

In this article, we approach this problem by applying formal methods to quality checking of medical guidelines. Here, we are mainly concerned with the meta-level approach [4] which we use to formalize general quality criteria of good practice medicine a guideline should comply to. A solid foundation can already be found in literature. In [2, 5] logical methods have been used to analyze properties of guidelines. In [4], it was shown that the theory of abductive diagnosis can be taken as a foundation for the formalization of quality requirements of a medical guideline in temporal logic. This result has been used in verifying quality requirements of good practice medicine of alternative treatments [3].

The contribution of this paper is twofold. First, we formalize quality requirements of medical guidelines which include, besides treatments, also the temporal relations between treatments. Second, us-

ing our quality requirements and medical background knowledge, we interactively verify a guideline dealing with the management of diabetes mellitus type 2. More specifically, we model the guideline as a ‘network of tasks’ using the language Asbru and, additionally, verify meta-level properties for this model in KIV, an interactive theorem prover. To the best of our knowledge, verification of a fully formalized guideline, as a network of tasks, using medical background knowledge has not been done before.

2 FORMALIZATION OF GUIDELINES

An example of a fragment of a guideline is shown in Figure 1, which is part of the guideline for general practitioners about the treatment of diabetes mellitus type 2, and is used as a running example in this paper. The guideline gives recommended interventions to be used for the control of the glucose level.

-
- Step 1: diet.
 - Step 2: if queetelet index (QI) ≤ 27 , prescribe a sulfonylurea drug; otherwise, prescribe a biguanide drug.
 - Step 3: combine a sulfonylurea drug and biguanide (replace one of these by a α -glucosidase inhibitor if side-effects occur).
 - Step 4: one of the following:
 - oral antidiabetic and insulin
 - only insulin
-

Figure 1. Guideline fragment on diabetes mellitus type 2 management. If one of the steps k is ineffective, the management moves to step $k + 1$.

It has been shown previously that the step-wise, possibly iterative, execution of a guideline can be described precisely by means of temporal logic [5]. In this paper we will use the variant of this logic supported by KIV [1], which is based on linear temporal logic, where time points are linearly ordered. For the verification of medical guidelines we assume at least three types of knowledge involved: medical background knowledge, order information from the guideline, and quality requirements.

Background knowledge: In diabetes mellitus type 2 various metabolic control mechanisms are deranged and many different organ systems may be affected. Glucose level control, however, is the most important mechanism. At some stage in the natural history of diabetes mellitus type 2, the level of glucose in the blood is too high (hyperglycaemia) due to decreased production of insulin by the B

¹ Institute for Computing and Information Sciences, University of Nijmegen, Netherlands, email: {arjehn, perry, peterl}@cs.ru.nl

² Institut für Informatik, Universität Augsburg, Germany, email: {balser, jonathan.schmitt}@informatik.uni-augsburg.de

cells. Oral anti-diabetics either stimulate the B cells in producing more insulin (sulfonylurea) or inhibit the release of glucose from the liver (biguanide). Effectiveness of these oral diabetics is dependent on the condition of the B cells. Furthermore, as a causal treatment, insulin can be prescribed. The mechanisms have been formalized in terms of a first-order predicate knowledge:

$$\text{knowledge} : \text{patient} \times \text{patient}$$

This predicate has been axiomatized with knowledge concerning the mechanism described above. For example

$$\begin{aligned} \text{knowledge}(\text{pre}, \text{post}) \rightarrow \\ (\text{insulin} \in \text{pre}[\text{'treatment'}] \rightarrow \\ \text{post}[\text{'uptake(liver,glucose)'}] = \text{up} \wedge \\ \text{post}[\text{'uptake(peripheral-tissue,glucose)'}] = \text{up}) \end{aligned}$$

denotes the physiological effects of insulin treatment. In this way, the predicate **knowledge** can be used to reason about the state transitions that occur during verification with temporal logic.

Medical guidelines in Asbru: The guideline fragment (Figure 1) was modelled in Asbru, as shown in Figure 2. This model consists of seven plans, which are ordered in a hierarchy. The top level plan **Treatments_and_Control** sequentially executes the four subplans **Diet**, **SU.or.BG**, **SU_and.BG**, and **Insulin.Treatments**. The latter is further refined by the two subplans **Insulin_and_Antidiabetics** and **Insulin**, which can be executed in any order.

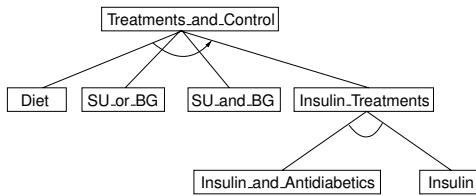


Figure 2. Asbru plan hierarchy of the diabetes mellitus type 2 guideline

Quality requirements: Here, we extend the formalization of good practice medicine of treatment choice [4, 3] to medical guidelines. Let \mathcal{B} be medical background knowledge, T be a treatment, P be a patient group, N be a collection of intentions, which the physician has to achieve, and M be a medical guideline. Then M is called a *proper* guideline according to the theory of abductive reasoning, i.e., $M \in \text{Pr}_P$, if:

- (M1) $\mathcal{B} \cup M \cup P \not\models \perp$ (the guideline does not have contradictory effects), and
- (M2) $\mathcal{B} \cup M \cup P \models \diamond N$ (the guideline eventually handles all the patient problems intended to be managed)

Furthermore, let \prec_φ be a partial order denoting a preference relation with $T \prec_\varphi T'$ meaning that T' is *more preferred* to T given criterion φ . If, in addition to (M1) and (M2), (M3) holds, with

- (M3) $O_\varphi(M)$ holds, where O_φ is a meta-predicate standing for an optimality criterion or combination of optimality criteria φ defined as: $O_\varphi(M) \equiv \forall M' \in \text{Pr}_P : \neg(M \prec_\varphi M')$,

then the guideline is said to be *in accordance with good-practice medicine*, denoted as $\text{Good}_\varphi(M, P)$.

3 VERIFICATION USING KIV

Using the interactive verification tool KIV, we have verified the quality requirements discussed above, written as a sequent with the initial state of a patient group, the initial state of the guideline, the medical guideline, effects of treatment plans, the background knowledge, and environment assumptions as assumptions. For example, we verified that the order of any two treatments in the guideline was consistent with the preference order \leq , which minimizes drugs and number of insulin injections:

$$\begin{aligned} \square \forall T (\text{Tick} \wedge T = \text{Patient}[\text{'treatment'}] \rightarrow \\ \rightarrow \square (\text{last} \vee (T \rightarrow \neg(T \leq \text{Patient}[\text{'treatment'}])))) \end{aligned}$$

Verification of such quality requirements could be done with a high degree of automation of upto 90%.

4 CONCLUSIONS

In our study we have setup a general framework for the verification of medical guidelines, consisting of a medical guideline, medical background knowledge, and quality requirements. We developed a model for the background knowledge of glucose level control in diabetes mellitus type 2 patients and a theory for quality requirements of good practice medicine based on the theory of abductive diagnosis. This model of background knowledge and theory of quality requirements were then used in a case study in which we verified several quality criteria of the diabetes mellitus type 2 guideline used by the Dutch general practitioners. In the case study we use Asbru to model the guideline as a network of tasks and KIV for the formal verification.

In the course of our study we have shown that the general framework that we have setup for the formal verification of medical guidelines with medical background knowledge is feasible and that the verification of quality criteria can be done with a high degree of automation. We believe both the inclusion of medical background knowledge and semi-automatic verification of the quality of decisions advised by the medical guideline are necessary elements for adequately supporting the development and management of medical guidelines.

REFERENCES

- [1] M. Balser, C. Duelli, W. Reif, and G. Schellhorn, ‘Verifying concurrent systems with symbolic execution’, *Journal of Logic and Computation*, **12**(4), 549–560, (2002).
- [2] J. Fox and S. Das, *Safe and Sound: Artificial Intelligence in Hazardous Applications*, AAAI Press, 2000.
- [3] A.J. Hommersom, P.J.F. Lucas, and M. Balser, ‘Meta-level Verification of the Quality of Medical Guidelines Using Interactive Theorem Proving’, in *JELIA’04*, volume 3229 of *LNCS*, pp. 654–666, Lisbon, Portugal, (2004). Springer-Verlag.
- [4] P.J.F. Lucas, ‘Quality checking of medical guidelines through logical abduction’, in *Proceedings of AI-2003, the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, eds., F. Coenen, A. Preece, and A.L. Mackintosh, volume XX, pp. 309–321, London, (2003). Springer.
- [5] M. Marcos, M. Balser, A. ten Teije, and F. van Harmelen, ‘From informal knowledge to formal logic: A realistic case study in medical protocols’, in *Proceedings of EKAW*, pp. 49–64. Springer, (2002).
- [6] A. Seyfang, R. Kosara, and S. Miksch, ‘Asbru’s reference manual, asbru version 7.3’, Technical Report Asgaard-TR-20002-1, Vienna University of Technology, Institute of Software Technology, (2002).

9. Planning and Scheduling

This page intentionally left blank

The Incompleteness of Planning with Volatile External Information

Tsz-Chiu Au and Dana Nau¹

Abstract. In many real-world planning environments, some of the information about the world is both external (the planner must request it from external information sources) and volatile (it changes before the planning process completes). In such environments, a planner faces two challenges: how to generate plans despite changes in the external information during planning, and how to guarantee that a plan returned by the planner will remain valid for some period of time after the planning ends. Previous works on planning with volatile information have addressed the first challenge, but not the second one.

This paper provides a general model for planning with volatile external information in which the planner offers a guarantee of how long the solution will remain valid after it is returned, and an *incompleteness theorem* showing that there is no planner that can succeed in solving *all* solvable planning problems in which there is volatile external information.

1 INTRODUCTION

In many real-world planning environments, the planner must request information from external information sources (databases, CAD systems, web services, and the like), incurring a lag time for receiving the answers. In practical planning situations, the planning activity may take more time than is needed to execute the resulting plan [6, 8, 9]. In such cases, some of the information on which the plan depends will change while the planning process is going on. For instance:

- [8] describes a domain-specific system that uses AI planning to do web service composition. Information from web services can change before planning finishes, and the system modifies its plans in response to such changes.
- At our university, the web site for booking concert-hall tickets displays a countdown timer showing how much longer the system will hold the seats that the user has booked. If the countdown ever reaches 0 (e.g., if user spends too much time examining the rest of the concert schedule), the booking will expire and the user will have to select seats all over again.
- When a traveler tries to construct a travel plan, the information about an airline flight may expire while the traveler was trying to plan some other details of the trip.

In such environments, how to cope with changes of external information and at the same time generate a plan that can be executed correctly is a big challenge. Most existing planners will not do this correctly unless they are modified.

In some cases, the interleaving of planning and execution [3, 7] is a good strategy to deal with volatile information.

But if the wrong choice of action can cause a failure that is irrecoverable (or recoverable only at a large cost), then it is necessary to reason, while the plan is being generated, about whether the assumptions that are being used to choose an action will still be true when the action is executed.

In [1] we described *query management strategies* that can adapt existing planners to deal with volatile external information by backtracking the planner to the first point in the planning process that makes use of outdated information, obtaining the up-to-date information, and resuming the execution from that point. A primary limitation of that work was that it provided no guarantee of completeness, i.e., no guarantee that a planner using such a query management strategy could succeed in every solvable planning environment.

In this paper, we provide an *impossibility theorem* showing that it is *impossible* to have a planner that can successfully find a valid solution in every solvable environment. This resembles the famous impossibility theorems in distributed systems (<http://www.podc.org/influential/2001.html>), which says that some problems in distributed systems are fundamentally unsolvable [2]. Remedy this problem is far from trivial, because it requires a modification to the fundamental assumptions of the distributed system model people have been used. Many of those solutions are not ideal, but they are needed in order to able to solve certain problems.

2 INCOMPLETENESS IN VEI-PLANNING

Let L be a planning language such as PDDL [5]. We construct a new language \hat{L} that contains all the symbols of L plus some additional symbols called *unknowns* as follows: given a set \hat{U} of unknowns, and a set $dom(u_i)$ of terms (called the *values* of u_i) in L for each unknown u_i , \hat{L} is a set of expressions produced by taking any expression e in L and replacing zero or more terms in e with corresponding unknowns.

For each unknown u , there is a piecewise-constant function $\mathbb{V}_u : \mathbb{R} \rightarrow dom(u)$, such that u 's value at time t is $\mathbb{V}_u(t)$. To learn about the value of u , a planner $\tilde{\mathcal{A}}$ must issue *queries* to an *information source* I_u : if $\tilde{\mathcal{A}}$ sends a query q for an unknown u to I_u , I_u will return a value $v(q) = \mathbb{V}_u(t')$ for q at time $t' = t_{return}(q)$. If $\tilde{\mathcal{A}}$ knows nothing about how long $v(q)$ will remain true, it cannot provide any sort of guarantee about the correctness of the plan it generates: the information may change during plan execution, invalidating the plan. Thus, we assume that I_u 's answer $ans(q)$ will include both $v(q)$ and an *expiration time* $t_{expire}(q)$. $v(q)$ is guaranteed to be valid until $t_{expire}(q)$, after which time it may change (see Fig. 1).

A *VEI-problem* \hat{P} (“*VEI*” stands for “volatile external information”) is a triple $\langle \hat{P}, T, \mathbb{E} \rangle$, where \hat{P} is a planning problem written in \hat{L} , T is a non-negative real number called a

¹ University of Maryland, College Park, U.S.A., email: {chiu,nau}@cs.umd.edu

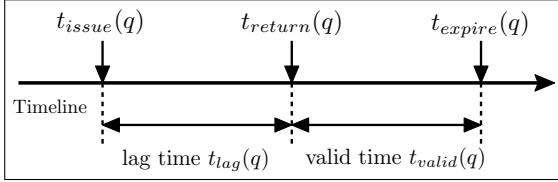


Figure 1. The lag time and the valid time of a query q . $v(q)$ is guaranteed valid only between $t_{return}(q)$ and $t_{expire}(q)$. After $t_{expire}(q)$, the value may possibly change.

validity guarantee, and \mathbb{E} is a (finite or infinite) set of VEI-environments, each of which is a pair $\langle \mathcal{E}_{lag}, \mathcal{E}_{ans} \rangle$, where $\mathcal{E}_{lag} : \hat{U} \times \mathbb{R} \rightarrow \mathbb{R}$ and $\mathcal{E}_{ans} : \hat{U} \times \mathbb{R} \rightarrow \hat{V} \times \mathbb{R}$ are functions giving the lag time and the answer to a query q for u issued at time t , respectively. We assume that a planner $\tilde{\mathcal{A}}$ resides in one of the VEI-environments in \mathbb{E} during its execution, but $\tilde{\mathcal{A}}$ has no prior knowledge about which one it is.

A value v of u is *confirmed* at time t if and only if $\tilde{\mathcal{A}}$ has issued a query q for u and has received an answer $ans(q) = (v, t_{expire}(q))$ such that $t_{return}(q) \leq t < t_{expire}(q)$. v is *T-confirmed* at time t if and only if v is confirmed at time t' for all $t \leq t' < t + T$. A solution π is *T-confirmed* at time t if all the values used by $\tilde{\mathcal{A}}$ for constructing π are *T-confirmed* at time t . A planner $\tilde{\mathcal{A}}$ *succeeds* in a VEI-problem $\langle \hat{P}, T, \mathbb{E} \rangle$ if and only if no matter which VEI-environment (among \mathbb{E}) $\tilde{\mathcal{A}}$ resides in, $\tilde{\mathcal{A}}$ returns a *T-confirmed* solution when $\tilde{\mathcal{A}}$ terminates. A VEI-problem \hat{P} is *solvable* if and only if there exists a planner that succeeds in \hat{P} . A planner $\tilde{\mathcal{A}}$ is *complete* if and only if $\tilde{\mathcal{A}}$ succeeds in *all* solvable VEI-problems.²

Theorem 1 states the necessary condition under which there exists a successful planner for a VEI-problem.

Theorem 1 A VEI-problem $\langle \hat{P}, T, \mathbb{E} \rangle$ is solvable only if for all $\mathcal{E} = \langle \mathcal{E}_{lag}, \mathcal{E}_{ans} \rangle \in \mathbb{E}$, (1) There exists a solution π for some instantiation of \hat{P} with values v_1, v_2, \dots, v_m for unknowns u_1, u_2, \dots, u_m , respectively; and (2) there exist times t_1, t_2, \dots, t_m such that the length of the time interval $\bigcap_{1 \leq j \leq m} [t_j + \mathcal{E}_{lag}(u_j, t_j), t_j + \mathcal{E}_{lag}(u_j, t_j) + t_{expire}^j] \geq T$, where where $\mathcal{E}_{ans}(u_j, t_j) = (v_j, t_{expire}^j)$.

For the above condition to be sufficient for solvability, the planner would need to be able to know instantaneously whenever a value expires. But that would require a communication channel with an infinite bandwidth and information sources with infinite amount of computational power—an impossible requirement. In practice, there is usually an unavoidable delay between two consecutive queries. For example, a database can only process one query at a time, and there is a delay between the processing of two consecutive queries. Therefore, it is realistic to make the following assumption:

Assumption 1 (Limited Accessibility) For each query q for an unknown u issued at time t , there is a period of time after t such that the planner cannot issue a query for u in the time period $[t, t + t']$, where t' is the length of the period.

Using this assumption, we can find two VEI-environments such that no planner can succeed in them at the same time.

² Our definition of completeness is the usual one for search algorithms [10, p. 71] and planning algorithms [4, p. 544]: an algorithm is complete if it finds a solution whenever a solution exists.

Lemma 1 Under the limited accessibility assumption, there exist two VEI-problems $\hat{P}_1 = \langle \hat{P}, T, \{\mathcal{E}_1\} \rangle$ and $\hat{P}_2 = \langle \hat{P}, T, \{\mathcal{E}_2\} \rangle$ that are solvable, but the combined VEI-problem $\langle \hat{P}, T, \{\mathcal{E}_1, \mathcal{E}_2\} \rangle$ is unsolvable.

This follows that there is no complete planner; otherwise, this lemma contradicts the fact that if both $\langle \hat{P}, T, \{\mathcal{E}_1\} \rangle$ and $\langle \hat{P}, T, \{\mathcal{E}_2\} \rangle$ can be solved by the same planner $\tilde{\mathcal{A}}$, the combined VEI-problem $\langle \hat{P}, T, \{\mathcal{E}_1, \mathcal{E}_2\} \rangle$ can also be solved by $\tilde{\mathcal{A}}$.

Theorem 2 (Incompleteness) Under the limited accessibility assumption, there is no complete planner.

3 CONCLUSIONS

Our incompleteness theorem shows that even if we restrict our attention to solvable VEI-environments, there is no planner that can solve every one of these environments. A consequence of this is that no planner dominates all other planners in terms of its coverage, i.e., there is no planner that can solve all of the environments that are solvable by other planners.

In future, we would like to study how to get around the incompleteness, so that one can create planners that are complete if certain restrictions are satisfied. One possible solution is to provide information about the timings of queries to planners, so that planners can schedule its queries in advance. Another solution is to look for randomized query strategies that have a high probability of success.

ACKNOWLEDGEMENTS

This work was supported in part by ISLE subcontract 0508268818 to DARPA's Transfer Learning program, UC Berkeley subcontract SA451832441 to DARPA's REAL program, and NSF grant IIS0412812.

REFERENCES

- [1] Tsz-Chiu Au, Dana Nau, and V.S. Subrahmanian, ‘Utilizing volatile external information during planning’, in *ECAI*, pp. 647–651, (2004).
- [2] Michael J. Fischer and Nancy A. Lynch, ‘Impossibility of distributed consensus with one faulty’, *JACM*, **32**(2), 374–382, (1985).
- [3] Michael R. Genesereth and Illah R. Nourbakhsh, ‘Time-saving tips for problem solving with incomplete information’, in *AAAI*, pp. 724–730, (1993).
- [4] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, May 2004.
- [5] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, ‘PDDL—the planning domain definition language’, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, (1998).
- [6] i2 Technologies. Reducing planning cycle time at altera corporation. <http://www.i2.com/assets/pdf/96FDF2C7-71C7-43B5-906A01BAE2F0AE76.pdf>, 2002.
- [7] Craig A. Knoblock, ‘Planning, executing, sensing, and replanning for information gathering’, in *IJCAI*, (1995).
- [8] Ugur Kuter, Evren Sirin, Dana Nau, Bijan Parsia, and James Hendler, ‘Information gathering during planning for web services composition’, *Journal of Web Semantics*, (2005).
- [9] W. H. McRaven, *Spec Ops : Case Studies in Special Operations Warfare: Theory and Practice*, Presidio Press, 1996.
- [10] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach (Second Edition)*, Prentice Hall, 2003.

Cost-Optimal Symbolic Planning with State Trajectory and Preference Constraints

Stefan Edelkamp¹

Abstract. State trajectory and plan preference constraints are the two language features recently introduced to PDDL in the context of the 5th international planning competition. For planning with soft constraints, an objective function monitors their violation. This paper introduces a symbolic approach for finding cost-optimal plans. The set-based branch-and-bound algorithm exploits an efficient symbolic representation of the objective function. State trajectory constraints are compiled into automata, while ordinary preferences are evaluated on-line for the intersection of the search frontier with the goal.

1 Introduction

Planning via Model Checking [2] exploits Boolean functions to lessen the costs associated with the exponential memory blow-up for the state sets involved as problem sizes get bigger. Given a fixed-length binary encoding for the state vector of a search problem, *characteristic functions* represent state sets. As the mapping is bijective, they are identified with the corresponding state sets. Transitions are formalized as relations, i.e., as sets of tuples of predecessor and successor states, or, equivalently, as the characteristic function of such sets. The *transition relation* T has twice as many variables as the encoding of the state. In action planning, we observe that T is the disjunction of all T_O with $O \in \mathcal{O}$ being an (instantiated) operator.

There are many different possibilities to come up with an efficient binary state encoding. The more obvious ones leads to bad performance of symbolic algorithms. Similar to [8], we infer a minimized finite domain encoding of a propositional planning domain.

2 Cost-Optimal Symbolic Search

For symbolic BFS, let Open_i be the set of states reachable from the initial state \mathcal{I} in i steps (initialized with $\text{Open}_0 = \mathcal{I}$) such that $\text{Open}_{i+1}(x') = \bigvee_{O \in \mathcal{O}} \exists x (T_O(x, x') \wedge \text{Open}_i(x))$. Note that S on the right hand side of the equation depends on x compared to x' on the left hand side. Thus, it is necessary to substitute x' with x in Open_i , written as $\text{Open}_i[x \leftrightarrow x']$. To terminate the exploration, we check if $\text{Open}_i \models \mathcal{G}$, with \mathcal{G} being the goal condition.

In order to retrieve the step-optimal plan we assume that all sets $\text{Open}_0, \dots, \text{Open}_i$ are available. We start with a state that is in the intersection of Open_i and \mathcal{G} . We take its characteristic function S into the relational product with T to compute its potential predecessors. Next we select the second last state on the solution path in the intersection of Pred and Open_{i-1} and iterate, until the entire plan has been constructed.

¹ Computer Science Department, University of Dortmund, Germany. Supported by DFG, projects Ed 74/2, Ed 74/3

Procedure Cost-Optimal-Symbolic-BFBNB

Input: Problem $\mathcal{P} = (\mathcal{I}, \mathcal{G}, \mathcal{O})$ with transition relations $T_O, O \in \mathcal{O}$
 Objective F (to be minimized on $[0, \max_f]$), Preferences X_p
Output: Cost-optimal plan from \mathcal{I} to state satisfying \mathcal{G}

```

 $U \leftarrow \max_f$ 
loop
     $Closed(x') \leftarrow \mathcal{I}(x');$   $Open(x) \leftarrow \mathcal{I}(x)$ 
     $Intersection(x) \leftarrow \mathcal{I}(x) \wedge \mathcal{G}(x)$ 
     $Bound(v) \leftarrow F(v) \wedge \bigvee_{i=0}^U [v = i]$ 
     $Eval(v, x) \leftarrow Intersection(x) \wedge \bigwedge_p X_p(v, x)$ 
     $Metric(x) \leftarrow \exists v : Eval(v, x) \wedge Bound(v)$ 
    while ( $Metric(x) \neq \perp$ )
        if ( $Open = \perp$ ) return "Exploration completed"
         $Succ(x') = \bigvee_{O \in \mathcal{O}} \exists x T_O(x, x') \wedge Open(x)$ 
         $Open(x) \leftarrow (Succ(x') \wedge \neg Closed(x'))[x' \leftrightarrow x]$ 
         $Closed(x') \leftarrow Closed(x') \vee Succ(x')$ 
         $Intersection(x) \leftarrow Open(x) \wedge \mathcal{G}(x)$ 
         $Eval(v, x) \leftarrow Intersection(x) \wedge \bigwedge_p X_p(v, x)$ 
         $Metric(x) \leftarrow \exists v : Eval(v, x) \wedge Bound(v)$ 
     $U \leftarrow ConstructAndStorePlan(Metric(x))$ 

```

Figure 1. Cost-Optimal BFS Planning Algorithm.

For the symbolic computation of a linear objective function $f(x) = \sum_{i=1}^n a_i x_i$, we first compute the minimal and maximal values (\min_f and \max_f) that f can take. This defines the range that has to be encoded in binary. For the ease of presentation we assume a cost-minimization problem, $x_i \in \{0, 1\}$, and $\min_f = 0$. The work of [1] implies that the BDD for representing f has at most $O(n \sum_{i=1}^n |a_i|)$ nodes and can be constructed with matching time performance. For preference constraints of type (*preference* $p \phi_p$) we associate a Boolean variable v_p (denoting the violation of p) and construct indicator relation $X_p(v) = (v_p \Leftrightarrow \phi_p)$.

Figure 1 displays the pseudo-code for a symbolic BFS-exploration incrementally improving an upper bound U on the plan cost. The search frontier denoting the current BFS layer is tested for an intersection with the goal condition, and this intersection is further reduced according to the already established bound. As there can be many different goal states contained in the remaining set, the plan construction process in *ConstructAndStorePlan* determines a state in the intersection that has the minimal cost. This is achieved by iterative intersection of $[v = i]$, for $i \in \{0, \dots, U\}$. The obtained value minus 1 is the return value for the next threshold.

Theorem The latest plan stored by the algorithm *Cost-Optimal-Symbolic-BFS* has minimal cost.

Proof The algorithm applies full duplicate detection and traverses the entire planning state space. It generates each possible planning state exactly once. Only clearly inferior states in *Intersection* are pruned when evaluated in *Eval* and taken into a conjunct with *Bound*. Therefore, *Metric* is trivial, only if there is no state in the intersection of the search frontier *Open* with the goal \mathcal{G} that has an improved bound.

3 Cost-Optimal Search with State Trajectories

State trajectory constraints can be interpreted Linear Temporal Logic (LTL) [7] and translated into automata that run concurrent to the search and accept when the constraint is satisfied [6]. LTL includes temporal modalities like A for *always*, F for *eventually*, and U for *until*. It is possible to compile the automata back to ordinary PDDL with each transition introducing a new operator [3]. Every automaton state for each automaton is represented by a proposition. For detecting accepting states we additionally include according predicate to the domain description. The initial state of the planning problem includes the start state of the automaton and a flag in case it is accepting. For all automata, the goal condition includes their acceptance.

As an example, for (*sometimes* ϕ) we apply automata-based model checking to build a (Büchi) automata for the LTL formula $F\phi$. Let \mathcal{S} be the original planning space and $A_{F\phi}$ be the constructed (Büchi) automaton for formula $F\phi$ and \otimes the synchronous cross product between two automata, then $\mathcal{P} \leftarrow \mathcal{P} \otimes A_{F\phi}$ and $\mathcal{G} \leftarrow \mathcal{G} \cup \{\text{accepting}(A_\phi)\}$. The initial state is extended by the initial state of the automaton, which in this case is not accepting.

For preferences for state trajectories that are constructed via automata theory, we apply the following changes. Instead of adding the automaton acceptance to the planning goal, we combine the acceptance with the violation predicate. If the automaton accepts then the preference is not violated; if it is located in a non-accepting state, then it is violated. For example, given (*preference p (at-most-once* ϕ) we explore the cross product $\mathcal{P} \leftarrow \mathcal{P} \otimes A_{\mathbf{A}\phi \rightarrow (\phi U(G \neg \phi))}$. Let $a = \{\text{accepting}(A_{\mathbf{A}\phi \rightarrow (\phi U(G \neg \phi))})\}$. A specialized operator *skip* allows to fail the automata constraint. Using a *dead* state, we assure that if an automaton is ignored once during the exploration, it remains invalid for the rest of the computation.

BDDs already save space for large state sets. However, even for symbolic search, additional memory can be saved. For example, Symbolic Branch-And-Bound Search seems to produce smaller search frontiers than Symbolic A* [9]. Our implementation features *Frontier-Search* [10], which has been proposed for undirected or directed acyclic graph structures. In the more general benchmark planning problems, a *duplicate detection scope* of 5 was sufficient to guarantee termination of *Cost-Optimal-Symbolic-BFBnB*. Moreover, it is not necessary to memorize an intermediate BDD layer corresponding to trajectory automaton transitions. Only the layers that correspond to the original unconstrained state space have to be stored.

The *variable ordering* has a large influence on the size of a BDD. We employ an interleaved variable ordering for the transition relation. Moreover, propositions that have been instantiated with the same domain objects should be encoded close together, while preference variables are better to be queried at the top of the BDD.

4 Experiments

Results on the approach are collected and published in the context of the 5th International Planning Competition. The following ta-

ble compares the outcome of the symbolic exploration for the domain *TPP/SimplePreferences* with our own explicit-state heuristic search planner on the competition machine, having a time-out (t.o.) at 1.800s, and a memory-out (m.o.) at 1 GB. The explicit-state planner runs out of time even in smaller problems, while the symbolic planner terminates if it has validated the optimal solution (label *). The symbolic planner runs out of memory in larger instances. In this case the solutions are optimal only wrt. the search depth reached.

Problem	Symbolic Search			Explicit Search		
	Time	Cost	Steps	Time	Cost	Steps
p01	1.34s	16*	17	t.o.	16	21
p02	1.68s	24*	14	t.o.	14	22
p03	2.45s	29*	20	t.o.	29	22
p04	6.06s	35*	20	t.o.	35	24
p05	337s	89	27	t.o.	223	24
p06	242s	110	21	t.o.	275	19
p07	246s	126	17	t.o.	322	19
p08	295s	141	17	t.o.	369	17

5 Conclusion

We have devised an optimal propositional PDDL3 [7] planning algorithm based on BDDs. As the approach for state trajectory constraints relies on a translation to LTL, it has the potential to deal with much larger temporal constraint language expressiveness than currently under consideration. As an alternative, Model checkers like SMV directly encode LTL formula into the transition relation without using an intermediate automaton [11]. We will likely extend the approach to domains with linear expressions also in the actions. As a prerequisite to apply [1], numerical state variables have to fit into finite domains. Many metric planning problems belong to this group. For more general planning problems we will consider using exploration for state sets represented in form of automata instead of BDDs [4].

REFERENCES

- [1] C. Bartzis and T. Bultan, ‘Efficient BDDs for bounded arithmetic constraints’, *STTT*, **8**(1), 26–36, (2006).
- [2] A. Cimatti, E. Giunchiglia, F. Giunchiglia, and P. Traverso, ‘Planning via model checking: A decision procedure for AR’, in *ECP*, pp. 130–142, (1997).
- [3] S. Edelkamp, ‘On the compilation of plan constraints and preferences’, in *ICAPS*, (2006).
- [4] J. Eisinger and F. Klaedtke, ‘Don’t care words with application to the automata-based approach for real addition’, in *CAV*, (2006).
- [5] M. Fox and D. Long, ‘PDDL2.1: An extension to PDDL for expressing temporal planning domains’, *Journal of Artificial Intelligence Research*, **20**, 61–124, (2003).
- [6] P. Gastin and D. Oddoux, ‘Fast LTL to Büchi automata translation’, in *CAV*, pp. 53–65, Paris, France, (2001).
- [7] A. Gerevini and D. Long, ‘Plan constraints and preferences in PDDL3’, Technical report, Department of Electronics for Automation, University of Brescia, (2005).
- [8] M. Helmert, ‘A planning heuristic based on causal graph analysis’, in *ICAPS*, pp. 161–170, (2004).
- [9] R. Jensen, E. Hansen, S. Richards, and R. Zhou, ‘Memory-efficient symbolic heuristic search’, in *ICAPS*, (2006).
- [10] R. E. Korf, W. Zhang, I. Thayer, and H. Hohwald, ‘Frontier search’, *Journal of the ACM*, **52**(5), 715–748, (2005).
- [11] V. Schuppan and A. Biere, ‘Shortest counterexamples for symbolic model checking of LTL with past’, in *TACAS*, pp. 493–509, (2005).

A Cooperative Distributed Problem Solving Technique for Large Markov Decision Processes

Abdel-Illah Mouaddib and Simon Le Gloannec¹

Abstract. In this paper, we consider the problem of solving a large MDP in a distributed way among several processors. To do that, we propose an approach which decomposes the large MDP into smaller ones each of which is solved on a unique processor. The obtained joint local policies derived from the small MDPs (subMDPs) behave in the same way of the policy of the initial MDP.

1 Introduction

The MDP provides a formal framework for modeling a large variety of stochastic planning and decision problems. The limitations of this framework are also well-known, where the most awkward for their application to realistic problems is the very large number of states. Many research lines have been developed using different (policy, space or action) decomposition techniques of MDPs [4, 3] to overcome this limitation. In this paper, we focus on a *parallel decomposition* where the construction, solving and execution of subMDPs are performed in a distributed way. Parallel decomposition planners in the literature consider simple interactions or an hierarchy structure. Our approach is a Parallel decomposition where subMDPs are connected as in a complete graph.

The concept of decomposition we consider is a problem formalized as a large MDP $\langle S, A, P, R, T \rangle$ that is decomposed into subproblems that can be further described by a subMDP of the initial MDP denoted by $M_i = \langle S_i, A_i, P_i, R_i, T \rangle$, where S_i is the state subspace, A_i is the action subspace, P_i is the probability transition in M_i , R_i is the reward function, and T the horizon. In this paper, we are interested in defining a mechanism for constructing subMDPs $M_i = \langle S_i, A_i, P_i, R_i \rangle$ from the global MDP $\langle S, A, P, R \rangle$ and deriving a global policy from local policies of subMDPs.

2 Decomposition of an MDP

The large MDP M is decomposed into a set of subMDPs (M_i) where each one is generated by a subset of actions A_i of the set A . All states generated by actions $A_{j \neq i}$ are cut and replaced by an abstract state that we name artificial state sa . We introduce, consequently, a null action aa that transitions to this abstract state. Also, states reachable at time t from the abstract state at time $t - 1$ (sa^{t-1}) are all states reached by actions of A_i from all states reached at time $t - 1$ by actions of the subsets $A_{j \neq i}$. These states that are represented abstractly with one state sa^{t-1} in the subMDP, should be communicated.

2.1 Decomposition of the action space A

The action space A is organized into a set of available actions at each time from 0 to T . A is decomposed into regions A_i s.t. : $A_i = \bigcup_{t=0}^{t=T} A_i^t$ and $A_i \cap A_{j \neq i} = \emptyset$ where A_i^t contains the available actions at time t of the region A_i (we consider any partition of A). Also, we add to each region the null action aa : $A_i = A_i \bigcup \{aa\}$

¹ GREYC-CNRS/Université de Caen, {mouaddib,sleglo} @info.unicaen.fr

where aa is considered to be available at any time. In the following, we describe, how from each region A_i , we construct a subMDP M_i .

2.2 State region S_i generation

The state region S_i for each subMDP M_i has an initial state s_i^0 . We consider that this initial state is the one of the original MDP s_0 . At step 0, all subMDPs M_i are initially at state s_0 : $\forall i, s_i^0 = s_0$. The generation of the region S_i consists in applying successively the actions of A_i^t followed by communication step from $t = 0$ to $t = T$.

S_i generation of M_i : return S_i

1. $reachable_i(0) = S_i = \{s_0\}$
 2. **for** $t = 1$ **to** T **do**
 3. $reachable_i(t) = \emptyset$
 4. **send** $reachable_i(t - 1)$
 5. **receive** $reachable_j \neq i(t - 1)$
 6. **for all** $s_i^{t-1} \in reachable_i(t - 1) \bigcup_{j \neq i} reachable_j(t - 1)$ **do**
 7. **for all** $a \in A_i^t$ **do**
 8. $reachable_i(t) = reachable_i(t) \bigcup \{transit(s_i^{t-1}, a)\}$
 9. $S_i = S_i \bigcup reachable_i(t)$
-

transit(s,a) generates states reachable from s by applying a . Let us note that $transit(s,aa) = \{sa\}$ the abstract state. Also, $transit(sa,a)$ generates states reachable from all states of M_j by applying action a . All these states should be communicated to the subMDP M_i . The cost of communication is that at each time t , subMDPs receive from the other a message containing the states generated at time $t - 1$. Therefore, the cost of generation is : $Cost_Gen = \sum_{i=1}^{i=n} \sum_{t=0}^{t=T} (n-1).|reachable_i(t)|$

3 Solving in Parallel subMDPs

The Bellman equation restricted to a subMDP M_i is then as follows:

$$V_i(s) = R_i(s) + \max_{a \in A_i} \sum_{s' \in S_i} P_i(s, a, s') V_i(s') \quad (1)$$

$$V_i(s_T) = R_i(s_T) \quad (2)$$

A policy π_i is a mapping from S_i to A_i assigning an action to each possible state in the subspace S_i . Solving this subMDP consists in deriving an optimal policy π_i^* s.t.:

$$\pi_i^* = argmax_{a \in A_i} \left(\sum_{s' \in S_i} P_i(s, a, s') V_i^*(s') \right) \quad (3)$$

We rewrite differently equation 1 in order to show more clearly the value of action aa : $V_i(s) = R_i(s) + \max_{a \in A_i - \{aa\}} \sum_{s' \in S_i} P_i(s, a, s') V_i(s')$ (4)

In Equation 4, we optimize over two terms: $P_i(s, aa, sa) V_i(sa) = V_i(sa)$ (aa is deterministic) and, $\max_{a \in A_i - \{aa\}} \sum_{s' \in S_i} P_i(s, a, s') V_i(s')$. These values are respectively of states reached by actions $A_{j \neq i}$ and the states reached by local actions $A_i - \{aa\}$ of a subMDP M_i . Stated differently, which first term is the value of state sa^t reached by action aa is obtained by communication while the values of states reached by local actions are computed locally.

3.1 Transitions and Value and Reward functions

- **Transition Probability:** The probability $P_i : S_i \times A_i \times S_i \rightarrow [0, 1]$. The probability P of the original MDP is given by $P : S \times A \times S \rightarrow [0, 1]$. Since $S_i \subset S$ and $A_i \subset A$, the transition probability $P_i(s, a, s')$ is s.t.: $P_i(s, a, s') = P(s, a, s')$, $P_i(s, \text{aa}, sa) = 1$ because there is no uncertainty on the fact that this subMDP does not execute any local action and $P_i(sa^t, a, s')$ is not needed because the value of the abstract state sa^t is communicated from $M_{j \neq i}$.
- **Values V_i and Rewards R_i :** Since $S_i \subset S$ then $R_i(s) = R(s)$. The reward function is not defined on abstract state sa^t because the value of these states are computed from equation 5.

Let us recall that the action **aa** is executed by a subMDP M_i when it expects that one of subMDPs $M_{j \neq i}$ can do better. Consequently, the value $V_i(sa)$ of state sa (reachable by action **aa** of M_i) represents the highest value that we can reward from the subMDPs $M_{j \neq i}$. Let the decision process at step t and s_j^t be the state at step t for any subMDP $M_{j \neq i}$ while subMDP M_i is in a reachable state s_i^t . The value $V_i(sa^t)$ at step t is the maximum of the values of states s_j^t reachable by an action $a \in A_j$. More formally :

$$\forall t, M_i \text{ and } M_{j \neq i} : V_i(sa^t) = \max_{j \neq i} V_j(s_j^t) \quad (5)$$

In this equation let s_j^t be the state of M_j corresponding to the state sa^t . The problem to solve is to select for each sa^t its corresponding states s_j received from subMDPs M_j . Then, to solve this equation, for each state sa_i^t , each subMDP M_i should receive from the other subMDPs $M_{j \neq i}$ all the values of their current state s_j^t . The subMDP M_i selects for each of its states sa_i^t the corresponding states s_j^t to solve equation (5). The interaction of subMDPs and value selection algorithms are given in the next sections.

3.2 Selecting appropriate values of a state sa_i^t

The objective of this section is the selection from the set of received values, the appropriate ones to compute the value of a given sa_i^t . At a step t , all subMDPs send values of the states s_j^t reached by one of its local actions ($\neq \text{aa}$). Not all states s_j^t are needed to compute the value of sa_i^t , but only the ones which are compatible with this state. The notion of *compatibility*, is defined using the notion of *sequence* of actions to reach a state and its duals.

3.3 Distributed Computation of values

The value computation algorithm starts by communicating the values of its terminal states for which $V_i(s^T) = R_i(s^T)$, while the value of states $V_i(sa^T)$ is computed using equation (5) where each $V_{j \neq i}(s_j^T)$ is sent from M_j to subMDPs $M_{k \neq j}$ including the subMDP M_i . In this way, all values at decision step T are known and allow subMDP M_i to compute the values of states at the previous decision step $T-1$. This processing combining the values communicated from the other subMDPs needed for equation (5) that in the same way is used to solve equation (4) is used backward from terminal states to the initial state for each subMDP. The value computation process uses a function **COM** which allows subMDPs to exchange their respective available information to compute all their values $V_i(sa_i^t)$.

3.4 Coordination of local policies

The local policies could be in conflict because only one action should be executed at a time as in the original MDP. To avoid the conflicts, the local policies should be able to select a local action when none of the other subMDPs can do better. The null action **aa** we added in each region A_i allows each local policy to do nothing (let the others do). Stated differently, at step t when the subMDP M_i decides to execute **aa**, one and only one of $M_{j \neq i}$ executes one of its local actions $a_j \neq \text{aa}$. The policy takes this decision when its value is greater than all the values rewarded by the local actions. To do that, the subMDP M_i needs to compute the value that we describe in the next section.

In the rest of the paper, we note sa_i^t by state reached by action **aa**, at step t , in subMDP M_i . Let us note that at step t , many states sa_i^t are generated which are distinguished by their parent state s^{t-1} .

Modified value iteration algorithm of M_i : return π

```

1. for all  $s_i^T \in \text{reachable}(T)$  do
2.    $V_i(s_i^T) = R_i(s_i^T)$ 
3.   COM( $M_{j \neq i}, V_i(s_i^T), \{V_i(sa_i^T)\}$ )
4.   for  $t = T-1$  down to 1 do
5.     for all  $s_t \in \text{reachable}(t)$  do
6.        $V_i(s_i^t) = R_i(s_i^t) + \max(V_i(sa_i^{t+1}), \max(\sum P_i(s_i^t, a, s_i^{t+1}) V_i(s_i^{t+1}))$ 
7.       COM( $M_{j \neq i}, V_i(s_i^t), \{V_i(sa_i^t)\}$ )
8.     for  $t = 1$  to  $T$  do
9.       for all  $s_i^t \in \text{reachable}(t)$  do
10.       $\pi(s_i^t) = \arg \max_{a \in A_i} V(\text{transit}(s_i^t, a))$ 

```

COM M_i (Communication between M_i and $M_{j \neq i}$): return $V_i(sa_i^t)$

```

1. send  $V_i(s_i^t)$  to  $M_{j \neq i}$ ;
2. receive from  $M_{j \neq i} V_j(s_j^t)$ ;
3. foreach  $sa_i^t$  do
4.   select from received values  $V_j(s_j^t)$ , values of compatible states  $s_j$ . Let these states be  $s_j^{sa}$ .
5. compute  $V_i(sa_i^t) = \max_{j \neq i} V_j(s_j^{sa})$  (equation 5)

```

4 Execution of subMDPs

In this section, we describe the execution of each subMDP M_i that follows its local optimal policy π_i^* . Let us recall that initially all the subMDPs are at the same state s_0 . At step t , each subMDP M_i is at a state s_i^t or sa_i^t . The decision made at this step is to select one of the actions $a \in A_i$ or **aa**. When the policy $\pi_i(s_i^t) \neq \text{aa}$, this means that all values received from the other subMDPs $M_{j \neq i}, V_j(s_j^t) < V_i(s_i^t)$. **Theorem 1** *The joint policy of subMDP is equivalent to the policy of the original MDP.*

Proof: contact the authors \square .

5 Costs of communication and computation

The decentralized way to perform MVIA uses a large amount of information exchanged via communication as shown in lines 3 and 7 of the MVIA. The cost of this communication is measured by the number of messages and the size of their contents. The cost of communication is of an order of magnitude of $O(n^2)$. This complexity can be reduced by using some specific architectures (router machine based on uni-cast or multi-cast technique) to linear complexity $O(n)$.

6 Discussion and Future work

Parallelizing the resolution of MDP is a hard problem that should consider three main issues : (1) completing the information lost during the decomposition, (2) solving in parallel decentralized MDPs and (3) monitoring the execution of local policies. These issues have been addressed by a new approach which allows us to construct from local policies a global optimal behavior. This approach can be seen as a distributed planning contribution of DEC-MDPs [1, 2] since most of algorithms are centralized planning approaches. Future work will concentrate on deep analysis of the type of MDPs and actions to better decompose the MDP or assuming some topologies of the network to make the parallelization more efficient.

REFERENCES

- [1] A. Beynier and AI Mouaddib, ‘A polynomial algorithm to solve decentralized mdp with temporal constraints’, in *AAMAS*, (2005).
- [2] A. Beynier and AI Mouaddib, ‘An iterative algorithm to solve constrained decentralized mdp’, in *AAAI*, (2006).
- [3] S. Le Gloannec, AI Mouaddib, and F. Charpillet, ‘A decision-theoretic scheduling of resource-bounded agents in dynamic environments’, in *ICAPS*, (2005).
- [4] R. Parr, ‘Flexible decomposition algorithms for weakly coupled markov decision process’, in *UAI-00*, (2000).

Integrating Off-line and On-line Schedulers

Riccardo Rasconi and Nicola Policella and Amedeo Cesta¹

Abstract. This work pursues an empirical study on the mutual interactions among a set of off-line and on-line constraint-based scheduling approaches. We devise a set of closed loop execution management algorithms, and compare their behavior within an experimental framework which allows to directly assess the consequences of each chosen strategy combination against the injection of a number of disturbing events, through simulated schedule executions.

1 SCHEDULE SYNTHESIS vs. EXECUTION

This work introduces an experimental platform which allows to compare different approaches to schedule synthesis and execution in a fair and controlled way. In this schema a schedule is initially produced off-line and thereafter delivered to an on-line module which takes care of assessing its dynamic characteristics by stressing it in a variety of ways, e.g. by simulating its execution under different environmental conditions.

Our aim is to reveal crucial “execution” aspects by plugging in different versions of either off-line solvers and/or on-line reschedulers. In particular the experimental framework is organized according to the following general conditions: (a) the solvers are chosen so as to produce initial solutions characterized by different levels of temporal flexibility; (b) the designated set of rescheduling algorithms generally follow a constraint-based approach; (c) the exogenous events set to disturb the execution of the schedules are exclusively of temporal nature (e.g., delays in the activities start times, lengthening in the activities durations, etc.).

In contrast to most off-line schedulers, which deliver fixed-time solutions, our approach is based on the general concept of “temporal flexibility” [4]. A temporally flexible solution can be described as a network of activities whose start times (and end times) are associated with a set of feasible values (feasibility intervals). Underlying the activity network there exists a Temporal Constraint Network (TCN [5]), composed of all the start and end points of each activity (time points), bound to one another through specific values which limit their mutual distances (activity-on-the-arc representation). The search approaches used in our schema focus on decision variables which represent conflicts in the use of the available resources; the solving process proceeds by ordering pairs of activities until all conflicts in the current problem representation are removed. This approach is usually referred to as Precedence Constraint Posting (PCP [4]), because it revolves around imposing precedence constraints (the *solution constraints*) on the TCN in order to solve the resource conflicts, rather than fixing rigid values to the start times.

In this analysis, schedules with different degrees of flexibility are initially generated. A certain level of temporal flexibility is guaranteed by a generic **Flexible Schedule (FS)**: a flexible schedule is a network of activities such that a feasible solution for the problem is obtained by allocating each activity at the temporal lower bound allowed by the network (in our experiments, flexible schedules are produced using the ISES procedure [3]).

In order to overcome the limitation imposed by flexible schedules, i.e. having only one consistent solution, a generalization of the TCN produced by a PCP phase is proposed in works such as [2, 7], in which methods for defining a set of both time and resource feasible solutions are presented. This new representation is called **Partial Order Schedule (POS)**: a Partial Order Schedule (*POS*) is an activity network, such that any possible temporal solution is also a resource-consistent assignment. A *POS* is a special case of a flexible solution and it can be obtained by replacing the solution constraints with a new set of constraints that impose a stronger condition on the TCN.

Regarding the on-line module, as soon as an event is acknowledged, a re-scheduling algorithm is invoked in order to produce an “updated solution” that will be used to continue the execution. In our schema, we model a local and global approach to rescheduling by enabling different constraint removal strategies on the current solution.² More specifically, we use a (1) *No-Retraction strategy (NR)* where all the constraints imposed are preserved, and the subsequent solution is computed by adding further constraints; and a (2) *Retraction strategy (R)* where, all the previously imposed solution constraints are removed and the subsequent solution is computed by adding new constraints to this “clean” representation.

Finally, regarding the exogenous events used to disturb the schedule execution, we limit ourselves to the generation of temporal events, such as delays of the activities start times (the activity a_i might undergo a delay of Δ_{st} time units, at $t = t_{\text{aware}}$) and/or modifications of activity processing times (the activity a_i ’s processing time p_i might be extended by Δ_p time units, at $t = t_{\text{aware}}$).

2 EXPERIMENTAL EVALUATION

The scope of this paper is to evaluate the behavior of schedules characterized by different temporal flexibility (**FS**, **POS**), as they are disturbed with a number of exogenous events, using different rescheduling strategies. The results of this analysis, never performed before, are presented in Table 1, and will be described in this section. The evaluation proposed below is based on the Resource-Constrained Project Scheduling Problem with minimum and maximum time lags, or RCPSP/max [1], a particular project scheduling problem which presents constraints that define the minimum and maximum distance between the execution of two activities.³

Eight different batteries are considered, where each of these is obtained coupling two scheduling problem benchmarks, *j30* and *j100* [6], with four execution simulations. The former consist of two sets of respectively 270 and 540 scheduling problem instances of different size, namely 30×5 and 100×5 (number of activities \times number of resources). On the other side, each execution simulation is composed of a set of properly modeled disturbing events: each scheduling problem is executed with four instances of world simulations of different size (1, 2, 3, and 5 events each) where each event represents either

¹ ISTC-CNR, Institute for Cognitive Science and Technology of the Italian National Research Council, email: name.surname@istc.cnr.it

² Exception is made for the constraints which model the dynamic aspects of the progressing execution. These constraints are, of course, always preserved.

³ RCPSP/max is a NP-hard problem. This is due to the presence of maximum time-lags, which imply the satisfaction of deadline constraints.

method	#d	j30					j100				
		mk	Δ mk	reschedulings	CPU off-line	CPU on-line	mk	Δ mk	reschedulings	CPU off-line	CPU on-line
FS-R POS-R FS-NR POS-NR fixed time	1	103.43	4.29	27.27%	4478.31	77.34	424.60	9.02	24.38%	52599.11	766.15
		102.63	3.60	5.19%	4613.57	15.13	419.88	5.07	11.58%	36287.57	303.97
		102.56	3.43	27.27%	4481.60	14.68	419.06	3.48	24.14%	36242.48	130.74
		102.14	3.10	5.19%	4612.60	2.34	417.11	2.31	11.58%	36251.86	54.59
		106.29	7.16	100.00%	4480.00	300.45	437.36	21.78	99.75%	67538.68	3035.68
FS-R POS-R FS-NR POS-NR fixed time	2	106.99	8.02	34.21%	4106.09	150.53	435.54	13.95	23.04%	30347.49	874.70
		104.91	6.03	4.51%	4242.03	20.23	429.22	8.41	10.03%	38529.15	674.86
		104.90	5.93	34.21%	4104.89	34.51	427.90	6.30	22.88%	30271.00	258.50
		104.83	5.95	4.51%	4239.40	3.83	424.74	3.93	9.56%	32371.82	97.46
		107.35	8.38	100.00%	4108.80	500.98	446.62	25.02	99.53%	30259.15	2768.37
FS-R POS-R FS-NR POS-NR fixed time	3	109.55	9.73	26.90%	4506.40	190.00	449.84	19.40	20.27%	26393.55	963.16
		108.11	8.36	5.85%	4647.54	37.02	441.49	11.93	9.41%	33716.81	675.18
		108.00	8.18	26.61%	4515.44	39.91	439.66	9.23	22.37%	26318.37	371.03
		107.83	8.09	5.85%	4646.23	7.72	436.24	6.68	9.63%	28406.98	151.26
		109.95	10.12	100.00%	4511.93	848.42	458.32	27.89	99.22%	26300.50	3716.15
FS-R POS-R FS-NR POS-NR fixed time	5	119.30	16.19	26.43%	4281.90	267.38	464.12	28.85	22.45%	25792.36	2391.92
		116.44	13.37	5.24%	4413.81	73.33	455.56	21.07	10.57%	33162.71	1748.56
		117.12	14.01	22.86%	4277.38	59.52	447.42	12.15	21.66%	25682.40	646.90
		115.86	12.79	5.00%	4410.60	12.14	444.68	10.18	10.48%	27544.98	289.17
		118.33	15.23	100.00%	4286.19	1161.19	465.56	30.29	98.43%	25754.32	6721.48

Table 1. Summarizing data for each execution strategy (computed on the intersection set of all successfully executed problems)

a delay on the start time, or a delay on the end time of the activities (these two different kinds of event are produced with the same probability).

The execution strategy combinations analyzed in this work are the FS-R, POS-R, FS-NR, and POS-NR, and are shown in Table 1. To make the comparison more complete, we add a further execution mode based on the use of fixed time solutions, where each activity is assigned a single start time instead of a set of alternatives.

For each execution policy, the table shows the average makespan at the end of every execution (mk), the average difference between the makespan at the beginning and at the end of every execution (Δmk), the number of performed reschedulings (*reschedulings*) related to the number of injected disturbances, the average CPU time necessary to compute the initial schedule (*CPU off-line*), and finally, the average CPU time which is necessary to perform all the reschedulings during the execution (all CPU times are expressed in milliseconds). For a fair comparison of the different policies, all the averages presented in this table are computed on the basis of the problem instances commonly executed with all the execution strategies.

A relevant result of this analysis is the extremely low rate of necessary reschedulings exhibited by the POS-R/POS-NR policies: this circumstance confirms the theoretical expectations which motivated the introduction of the \mathcal{POS} s. As shown, the need for schedule revision in case of \mathcal{POS} utilization decreases by more than 75% in case of the j30 set, and by at least 50% in case of the j100 set with 5 disturbances. It is worth noting as this result is confirmed by the better makespan and CPU on-line values in the case of \mathcal{POS} s. We note also the $\approx 100\%$ *reschedulings* figure relative to the case of *fixed time* schedules where repairs are practically always needed. This case reveals the highest makespan elongation (see mk and Δmk) and *CPU on-line* values: in fact, the frequent rescheduling actions inevitably tend to spoil makespan quality.

Another interesting aspect is related to the comparison of the *CPU on-line* values between the *Retraction* and *No-Retraction* strategies: NR strategies reveal lower CPU-load rates with respect to the R strategies, *despite the comparable amount of performed reschedulings*. In fact, NR execution modes retain all the temporal constraints of the previous solution: hence, the rescheduler is bound to work on a smaller search space, finding the next solution almost immediately. Additionally, the tremendous on-line CPU load in the case of fixed time solutions, as well as how the reactivity of NR technique, combined with the low revision requirements featured by the \mathcal{POS} , allows the fastest dynamic reactions of the whole set (see the *CPU on-line* values for POS-NR).

We conclude mentioning that in further analysis we have also discovered some limitations in using \mathcal{POS} s with respect to the ability

of completing the schedule execution. We refer the reader to [8] for a more detailed presentation of these issues.

3 CONCLUSIONS

This paper discusses alternative approaches to the execution, monitoring and repair of scheduling solutions. We obtained different insights on the combination of two aspects: (a) the use of temporal flexible solutions to increase proactive robustness and (b) the use of complementary re-scheduling policies to react to unexpected events. In particular we analyze two alternatives to the classical fixed time schedule – flexible schedules, containing a single point solution, and partial order schedules or \mathcal{POS} s. Moreover, we distinguish between the incremental modification of the initial schedule vs. the retraction of previously made decisions followed by a new resolution. The empirical results support the usefulness of this analysis and represent a direct confirmation of theoretical expectations. For the first time, the positive effects of temporal flexibility in the definition of temporal solutions are emphasized. This represents a clear starting point for further investigations and opens the ways to new perspectives.

Acknowledgments. The authors' work is partially supported by MIUR (Italian Ministry for Education, University and Research) under project "Vincoli e Preferenze" (PRIN).

REFERENCES

- [1] M. Bartusch, R. H. Mohring, and F. J. Radermacher, 'Scheduling project networks with resource constraints and time windows', *Annals of Operations Research*, **16**, 201–240, (1988).
- [2] A. Cesta, A. Oddi, and S. F. Smith, 'Profile Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems', in *Proceedings of AIPS-98*, pp. 214–223. AAAI Press, (1998).
- [3] A. Cesta, A. Oddi, and S. F. Smith, 'An Iterative Sampling Procedure for Resource Constrained Project Scheduling with Time Windows', in *Proceedings of IJCAI-99*, pp. 1022–1029. Morgan Kaufmann, (1999).
- [4] C. Cheng and S. F. Smith, 'Generating Feasible Schedules under Complex Metric Constraints', in *Proceedings of AAAI-94*, pp. 1086–1091. AAAI Press, (1994).
- [5] R. Dechter, I. Meiri, and J. Pearl, 'Temporal Constraint Networks', *Artificial Intelligence*, **49**, 61–95, (1991).
- [6] R. Kolisch, C. Schwindt, and A. Sprecher, 'Benchmark Instances for Project Scheduling Problems', in *Project Scheduling - Recent Models, Algorithms and Applications*, ed., J. Weglarz, 197–212, Kluwer Academic Publishers, Boston, (1998).
- [7] N. Policella, A. Oddi, S. F. Smith, and A. Cesta, 'Generating Robust Partial Order Schedules', in *Proceedings of CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pp. 496–511. Springer, (2004).
- [8] R. Rasconi, N. Policella, and A. Cesta, 'Fix the schedule or solve again? comparing constraint-based approaches to schedule execution', in *Proceedings of ICAPS workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, (2006).

Environment-Driven Skeletal Plan Execution for the Medical Domain

Peter Votruba¹, Andreas Seyfang¹, Michael Paesold¹, Silvia Miksch^{1,2}

Abstract. An important application of both data abstraction and plan execution is the execution of clinical guidelines and protocols (CGP), both to validate them against a large set of test cases and to provide decision support at the point of care. CGPs can be represented and executed as a hierarchy of skeletal plans. To bridge the gap between low-level data and high-level concepts in the CGP, intelligent temporal data abstraction must be integrated with plan execution.

In this paper we describe a solution to this challenge which was implemented as part of the European project Protocure II to improve the quality of CGPs. They are translated to the high-level plan representation language Asbru which again is compiled into a network of abstraction modules by the system. Then this network performs the content of the plans triggered by the arriving patient data.

By this, we seamlessly integrate the synchronisation of guideline execution with observed patient state, complex temporal abstractions and execution of complex plans without requiring the user to handle the low-level details. Instead, user-friendly tools are used to create and maintain the guideline.

1 Introduction

In the field of medicine, the application of clinical guidelines and protocols helps to improve the quality of care by ensuring the optimal choice of treatment. A precondition for the successful application of clinical guidelines and protocols is the automatic abstraction of context-dependent time-annotated raw-data (e.g., percent of oxygen in blood at a certain second) to high-level medical concepts (e.g., sufficient oxygen saturation during an extended period of observation). This is performed by temporal data abstraction.

Within the domain of medicine, intensive care poses additional challenges to plan execution. The nature of the data raise the need for elaborate pre-processing of the data, including complex knowledge-based plausibility checks and correction heuristics which involve multiple channels and time windows and the synchronisation of low-frequency and high-frequency inputs. This processing together with the sometimes complex propagation of plan states needs to be performed in a timely manner to meet the real-time requirements of online patient monitoring.

To meet these requirements, we integrate time-oriented, skeletal planning using the Asbru representation with real-time monitoring and temporal data-abstraction.

There are many guideline modelling approaches today [4, 6, 1], but only few integrate strong data abstraction resources. This may

be caused by the fact that in most settings, data is entered manually which allows to delegate the data abstraction task to the user by demanding qualitative high-level input instead of the original data. However, the integration of guideline execution into the clinical data flow becomes more and more important in order to apply decision support systems in clinical daily practice [2].

2 Plan Representation in Asbru

Asbru is a time-oriented plan representation language that represents clinical guidelines as skeletal plans [5]. Asbru's distinguishing features are that intentions, conditions, effects and world states have a temporal dimension and are continuous (durative). Because of the advanced temporal data abstraction capabilities, diagnosis and treatment can be tightly integrated allowing each one to support the other one.

All conditions for the transition from one plan state to another are expressed in terms of temporal patterns. A temporal pattern consists of one or more parameter propositions or plan-state descriptions. Each parameter proposition contains a value description, a context, and a time annotation. The time annotation used allows a representation of uncertainty in starting time, ending time, and duration of an interval. Start and end are defined as shifts from a reference point. Reference points can be defined as sets of cyclical time points or references to parameter changes, allowing repeated temporal patterns.

3 The Asbru Interpreter

Conceptually, the Asbru Interpreter consists of three basic units: data abstraction, monitoring, and plan execution. In the data abstraction unit, various temporal or atemporal abstractions are applied to the patient data to gain information at higher conceptual levels. The provided quantitative or qualitative data is monitored to detect temporal patterns in the abstracted data. This information is used to control the selection and execution of plans. This data flow is not unidirectional, instead, the execution unit can interact with both monitoring and abstraction unit to adjust the monitored patterns and to adapt the abstraction process to the context given by the current plan states.

The interpreter has two different modes of operation: Batch mode and interactive mode. In batch mode, a large set of records is read to validate a guideline against patient data or to create complex abstractions of the data for later analysis. In interactive mode, data can be read from monitoring devices in addition to the user input.

Figure 1 shows the parts of the interpreter on the implementation level. The main parts are the Asbru Compiler and the Execution Manager. The three conceptual components mentioned above – data abstraction, monitoring, and plan execution – are seamlessly integrated in the module graph.

¹ Institute of Software Technology, Vienna University of Technology, Austria

² Department of Information & Knowledge Engineering, Danube-University Krems, Austria

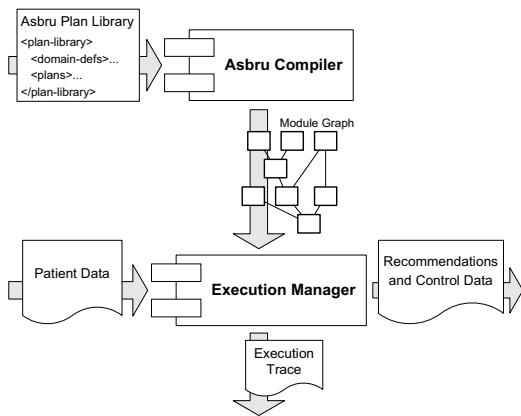


Figure 1. System architecture and data flow in the Asbru Interpreter. The Asbru plan library is compiled into a directed graph of modules by the Asbru Compiler. The Execution Manager uses this module network to process patient data and execute the plans representing the guideline.

At program start-up, the Asbru plan library XML file is compiled into a directed graph of modules.

For each time step, the Execution Manager enacts each of the modules in the network to process patient data, monitor temporal patterns, and execute the plans in the guideline. These modules are largely compatible with each other, which allows information extracted by any module to flow back into the abstraction or monitoring process. To handle complex networks with many inputs in high-frequency applications, the Execution Manager ensures that each module is enacted exactly when needed, allowing for small time steps by some modules without the overhead created by other modules which would not provide new information at that moment.

Modules can set alarms, to be triggered when a certain span of time is elapsed. Here we distinguish between *pre-alarms* and *post-alarms* depending on whether the alarm is triggered before or after processing the data for this time step. This distinction allows the implementation of both complex and convex temporal intervals. Alarms are set by various monitoring and value aggregation modules.

The available modules can be divided into the following categories.

Raw data modules. These modules interface the input channels and map the raw-data parameter definitions in Asbru.

Value abstraction modules. This group comprises the logical and arithmetic combination of inputs, and the mapping of quantitative values to qualitative categories.

Value aggregation modules. In order to map high-frequency, error-prone inputs to high-level concepts, it is mandatory to aggregate series of measurements and to derive the abstractions from them, and not from single measurements.

Monitoring modules. These modules handle temporal patterns, such as parameter propositions, which control the state changes of Asbru plans.

Temporal abstraction modules. The patterns detected by monitoring modules and aggregates of the measurements often need one or more steps of temporal abstraction to detect complex patterns in the input data such as "five episodes of apnea followed by hyperoxemia during the previous hour".

Plan modules. Modules in this category represent Asbru plans or single plan steps. The network of parent and child plans is fully integrated with the other modules to form the module graph.

4 Evaluation and Future Work

An extensive real-world guideline [3] was modelled in Asbru and test runs of the resulting model in the interpreter were successful. Running the interpreter on patient cases will allow the comparison between the expected outcome of applying the guideline and the actual outcome according to the model in Asbru.

In practical tests with high-frequency data recorded at an intensive care unit, the interpreter processed input from 10 channels and moderately complex abstractions thereof at a rate of more than 1 kHz on a standard PC. Most clinical data is recorded at 1 Hz, or 200 Hz. We therefore conclude that the computational performance is sufficient for real-time applications in clinical monitoring.

Future work will go into the construction of dedicated modules to interface equipment at intensive care units, where it will be employed in clinical studies. In addition, a graphical user interface to allow the interactive use of the interpreter by non-computer experts is under development.

5 Conclusion

Plan execution in real-world high-frequency domains such as intensive care units demand for tight integration of temporal data abstraction and plan execution to achieve the required intelligent reaction to unpredictable changes in the environment, i.e., the patient state.

While the knowledge in such domains is abstract, partly vague or incomplete, and often complex, the data arrives at a high rate and in a format that is far from the level in which the domain knowledge is specified. Translating the domain knowledge to such low levels by a knowledge engineer leads to well known short-comings regarding maintenance and assuring the correctness of the model.

We therefore designed an interpreter, which takes a high-level specification of skeletal plans and temporal data abstraction and compiles them into a network of abstraction modules. Using elaborate management of data flow, these modules process the data at a high rate, even in complex configurations.

Tests have demonstrated that the interpreter can handle a complete guideline, large sets of patient records, and high-frequency measurements. Applications to a large set of data are currently in progress.

ACKNOWLEDGEMENTS

This work is part of the Procure II project, which is supported by the European Commissions IST program, under contract number IST FP6-508794.

REFERENCES

- [1] P. Ciccarese, E. Caffi, L. Boiocchi, S. Quaglini, A. Kumar, and M. Stefanelli, 'A guideline management system', in *MedInfo 2004*, ed., M. Fieschi et al., pp. 28–32, Amsterdam, (2004). IOS Press.
- [2] W. Horn, 'AI in medicine on its way from knowledge-intensive to data-intensive systems', *Artif Intell Med*, **23**(1), 5–12, (2001).
- [3] Nationaal Borstkanker Overleg Nederland (NABON), *Guideline for the treatment of breast carcinoma*, Van Zuiden Communications B.V., 2002.
- [4] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and M. Stefanelli, 'Comparing computer-interpretable guideline models: A case-study approach', *JAMIA*, **10**(1), (2003).
- [5] A. Seyfang, R. Kosara, and S. Miksch, 'Asbru 7.3 reference manual', Technical report, Vienna University of Technology, (2002).
- [6] P. Terenziani, G. Molino, and M. Torchio, 'A modular approach for representing and executing clinical guidelines', *Artif Intell Med*, **23**(3), 249–276, (2001).

Time constrained VRP: an agent environment-perception model

Mahdi Zargayouna¹

Abstract. In this paper, we present a multiagent model in which agents have a perception upon their shared environment, a measure is associated to the agents' perception field. We apply the model on the Vehicle Routing Problem with Time Windows (VRPTW). The overall process adopts the general schema of parallel insertion methods and it uses the contracting of perception's field of the vehicle agents as a new distance between them. This new measure expresses the feasibility universe of the vehicles and is used as a criterion of choice between candidates vehicles for the insertion of a customer in their plan. Our approach provides a new method to tackle the Time constrained VRP in which the solving process is focused on the future and constitutes an alternative for handling the dynamic version of the problem.

1 Introduction

Vehicle routing problems (VRP) have been an intensive research area in the last decades because of their large applicability in real life problems. Several constrained variants were proposed in order to meet specific operational applications, they concern time restrictions, requests configuration etc. One of the most widely studied problem is the time (and capacity) constrained version: the Vehicle Routing Problems with Time Windows (VRPTW henceforth). VRPTW and their variants (PDPTW, DARPTW etc.) are hard combinatorial optimization problems. Exact approaches for the static VRPTW are only of theoretical interest and artificial intelligence heuristics succeeded in finding good solutions, within reasonable computational times. In the dynamic problem, the whole customers set is not available before the start of execution.

In this paper, in order to tackle the dynamic VRPTW, we consider that zero demands are known *a priori*. To this purpose, we propose to model a multiagent system (MAS henceforth) composed of *vehicle* agents, a queue of *customer* agents and events representing customers' requests occurring *ad hoc*. They evolve in the same environment in which they are described by their observable descriptors. *Vehicle* agents' perception field is modeled as a subpart of the environment observation domain and is quantified. *Customer* agents select the vehicle into which they want to belong following a criterion: the minimal decrease of the *vehicle* agent's perception field quantification. This criterion helps us in dealing with uncertainty concerning future (unknown) customers' demands. The remainder of the paper is then organized as follows. In the next section, we first present a generic model of the environment and of the agents' perception, then its application to the VRPTW problem. In section 3, we conclude and give some perspectives to our work.

¹ Inrets institute and Paris-Dauphine University, France, email: zargayou@inrets.fr

2 Our proposal

2.1 Multi-Agent environment and Agents' perception

Our environment model could be seen as an extension of [1, 6]. Our definition is based on the principle of observability i.e. the entities' external description. The formulation is inspired from the Symbolic Data Analysis paradigm [2].

Definition 1 (Environment) $ENV = \langle A, E, Y, M \rangle$ with:

- $A = \{a_1, \dots, a_m\}$ a set of agents,
- $E = \{e_1, \dots, e_n\}$ a set of events,
- $Y = \{y_1, \dots, y_p\}$ a set of variables (functions)

$$Y : A \cup E \rightarrow \Omega$$

$$\Omega = \prod_{i=1}^p \Omega_i, \text{ with } \Omega_i \text{ the description domain of the variable } y_i,$$

$$\bullet M : \prod_{i \in I} \Omega_i \rightarrow \mathcal{P}(A), I \subseteq \{1..p\}$$

In addition, to every agent a_k , we associate:

$$Restr_k : \prod_{i \in I_k} \Omega_i \rightarrow \{\text{true}, \text{false}\}, I_k \subseteq \{1..p\}$$

A quantification: $\| a_k \| \in \mathbb{R}^+$

Every agent and every event are accessible via their observable descriptions (set of variables). To every agent a_k , we associate a conjunction of perception restrictions (assertions, we note $Restr_k$) on the values of the variables it wants to perceive (I_k is defined by a_k). If $Restr_k$ is empty then a_k can perceive everything. To every $Restr_k$ of an agent a_k , we associate a quantity in $\mathbb{R}^+ (\| - \|)$ which reflects the capacity of perception of agent a_k , based on its restrictions $Restr_k$. For instance, if $Y = \langle \text{age}, \text{sex} \rangle$, $\Omega = [0, 100] \times \{\text{"male"}, \text{"female"}\}$, and $Restr_1 = \langle \text{sex} = \text{"male"} \wedge \text{age} \geq 50 \rangle \Rightarrow \| a_1 \| = 1 \times 50 = 50$.

The environment is responsible of the aggregation of the different $Restr$ of the agents -via the function M - in order to know, when an agent or an event reveals to the system (partially defined by $\bigcup_{i \in I} y_i$ with $I \subseteq \{1..p\}$), which agent actually perceives it.

Analogically to the Symbolic Data Analysis paradigm, every function $Restr_k$ is a symbolic object which extension is composed of $Ext_k = \{\omega / Restr_k(\omega) = \text{true}\}$.

Such an abstraction of the environment can model any MAS where the environment is a first-class abstraction [5] i.e. the environment is an explicitly designed entity. Note that to our knowledge, in the domain of the modeling of environment for multiagent systems, there is no propositions in the literature that quantify the agents' perception.

2.2 Application: the VRPTW problem

The VRPTW problem is a multi-vehicle Traveling Salesman Problem with limited capacity and with time intervals associated to the nodes to be visited. We consider an Euclidean VRPTW for presentation clarity (distances and time-travel computed following the Euclidean metric). In our system, we have two types of agents: *customer* agents and *vehicle* agents. *customer* agents are in a queue, they sequentially reveal to the system and each one is considered independently from its successors. Every *customer* agent is described by (x, y, e, l, s, q) , with (x, y) its cartesian coordinates, $[e, l]$ its time window, s its service time, and q its requested quantity. Each *vehicle* agent is described by cap its capacity and is initially located at a depot. The overall process is greedy-like and is based on a CNP (*Contract Net Protocol* [3]), where every *vehicle* agent that is able to visit the considered customer (w.r.t the customers already inserted in its plan) proposes an offer, and the best offer (following a criterion that we defined, see below) is retained by the *customer* agent. In order to be perceived by the *vehicle* agents, a *customer* agent, when it is its turn to reveal to the system, creates events representing it by discretizing its time window $[e, l]$ and puts them in the environment, every event is then described by (x, y, s, t, q) . The offer computed by every *vehicle* agent is equal to its contracting of perception i.e. the decrease of its perception quantification as presented in the model. The initial restrictions, say *Restro* of a *vehicle* agent, say v_0 , following the Euclidean metric and the capacity constraints, are equal to:

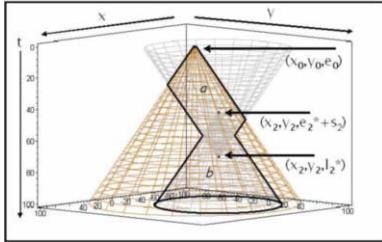


Figure 1. Space-time perception

$$Restro_0 = ((x - x_0)^2 + (y - y_0)^2)^{1/2} \leq (t - t_0) \wedge (q \leq cap_0)$$

(x_0, y_0, t_0) are the cartesian coordinates of the depot and the scheduling starting time (constants) and cap_0 the initial capacity of the vehicle v_0 . From the point of view of an agent, following this inequality, the environment is a hypercube of four dimensions (two for the cartesian coordinates x, y , one for the time t and one for the capacity q). Initially, $\|v_0\|$ is equal to the volume of the cone of vertex (x_0, y_0, t_0) (cf. Figure 1, the big cone) multiplied by cap_0 . Now, let us consider a *customer* agent c_2 described by $(x_2, y_2, e_2, l_2, s_2, q_2)$. Suppose that v_0 perceived a subset of the events describing it. v_0 computes the new time-window values e_2^* , l_2^* ², and computes its new *Restro* and $\|v_0\|$. The new *Restro* is equal to $[a] \vee [b]$ with:

$$[(x - x_2)^2 + (y - y_2)^2]^{1/2} \leq (l_2^* - (t + s)) \quad [a]$$

$$[(x - x_2)^2 + (y - y_2)^2]^{1/2} \leq (t - (e_2^* + s_2)) \quad [b]$$

[a] is the condition to be satisfied by any customer if it had to be inserted before c_2 and [b] is the condition it has to satisfy if it had to be inserted after c_2 . The new quantification $\|v_0\|$ is equal to the

² how to recompute e_2^* and l_2^* is out of the scope of this paper

volume of the intersection of the old cone with the union of the two new cones described by [a] and [b] (cf. Figure 1), multiplied by the new cap_0 ($= cap_0 - q_2$). v_0 is a candidate for the insertion of c_2 , the offer it will propose is $\Delta_{\parallel - \parallel}$, the difference between its old and its new perception quantification, which is equal to $\|v_0\|_{new} - \|v_0\|_{old}$. c_2 selects the vehicle with the minimum loss of perception $\min \Delta_{\parallel - \parallel}$. This criterion expresses the decrease of the probability for a vehicle to be a candidate for future customers' demands.

We ran our system on the Solomon's benchmark [4]. Although our system could not actually be compared to any state-of-the-art works, because our proposal is the only one that could work with zero known demands, we compared it with the best known static systems focusing on the verification of two results: the computational times' variation and the explosion of the number of vehicles used. The results showed that the computational times increase slowly with respect to the number of considered customers, which makes our proposal a good candidate to operate in a dynamic configuration. The gap in the number of vehicles used by our system w.r.t those used by the best algorithms is of 50% when considering 100 customers and 16% when considering 200 customers, thus the fleet of vehicles remains stable even assuming this handicapping scenario.

3 Conclusion and perspectives

In the state-of-the-art approaches, all the works try to handle the dynamic requests in the VRPTW by, firstly, considering a subpart of requests known in advance. Secondly, some of them try to reconsider previous requests assignment in order to increase the quality of the solution, which is time consuming. We try to follow neither the first nor the second approach: no customers are known in advance (the newly incoming ones simply enqueue themselves), and every assignment is definitive. We present a generic multi-agent model based on the observability principle, we quantify agents' perception and we use the loss of perception as a new distance between partial *vehicle* agents' plans in order to lessen the myopic behavior of the traditional metrics. This model provides a new method to tackle the dynamic VRPTW in which the solving process is focused on the future. Future works will consider variants of the VRPTW such that the Pickup and Delivery Problem with Time Windows and the Dial-A-Ride problem, the latter problem fits very well with the multiagent paradigm since customers also try to maximize their own utility.

REFERENCES

- [1] Balbo, F., Pinson, S.: "Toward a Multi-Agent Modelling Approach for Urban Public Transportation Systems", in Omicini A., Petta P. et Tolksdorf R. (eds), Engineering Societies in the Agent World II, LNAI 2203, Springer Verlag (2001) pp 160-174.
- [2] Bock, H.H., Diday, E. (eds.): "Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data", Series: Studies in Classification, Data Analysis, and Knowledge Organisation, Vol. 15, Springer-Verlag, Berlin, Germany (2000), 425 p.
- [3] Davis, R., and Smith, R.G. : "Negociations as a metaphor for distributed problem solving. Artificial Intelligence 20:63-109, 1983.
- [4] Solomon M. : "Algorithms for the vehicle routing and scheduling with time window constraints", in Operations Research 15, (1987), pp 254-265.
- [5] Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: "Environments for Multiagent Systems, State-of-the-art and Research Challenges", in Proceedings of Workshop on Environments for Multi-Agent Systems (E4MAS) LNAI 3374 Springer Verlag (2004) pp 1-47.
- [6] Zargayouna, M., Balbo, F. and Saunier, J.: "Agent Information Server: a middleware for traveler information", In Lecture Notes of Artificial Intelligence, Vol.3963, 2006.

10. Robotics/Perception

This page intentionally left blank

On Packing 2D Irregular Shapes

Alexandros Bouganis and Murray Shanahan¹

Abstract. Designing and implementing an intelligent system that tackles the problem of placing two-dimensional shapes on a surface, such that no shapes overlap and the uncovered surface area is minimized, is highly important in industrial applications. However, it is also interesting from the scientific perspective, in terms of artificial intelligence, since autonomous systems developed up to now have found it difficult to compete with humans in this task. This paper presents a new algorithm which addresses the on-line packing of two-dimensional irregular shapes, and achieves high quality solutions in short computational times. The key point of this algorithm is the utilization of techniques drawn from computer vision and artificial intelligence.

1 Introduction

The two-dimensional *packing problem* can be defined as the problem of placing a number of two-dimensional shapes on a surface, such that no shapes overlap and the uncovered area of the surface is minimized. This problem arises in a variety of industrial applications within shipbuilding, textile, wood, plastic, sheet metal and leather manufacturing. However, this problem is not only interesting for industrial reasons, but also from an artificial intelligence perspective, since algorithms that have been developed up to now have found it difficult to compete with humans [7].

Two variations of the problem are *on-line* and *off-line* packing. In off-line packing, the shapes of *all* the parts to be placed are given *a priori* to the packing algorithm. In this way, the packing algorithm enjoys the flexibility of placing the parts in a sequence that it considers best. Contrary to off-line packing, in the on-line instance of the problem, the sequence of the parts to be placed is *randomly* generated and the packing algorithm acquires *only* the part that it has to place at each time, and does not have *a priori* knowledge of the parts that has to place in the future. Obviously, in on-line packing it is much harder to achieve high quality solutions than it is in the off-line case. The present paper introduces an intelligent system that tackles the on-line packing problem effectively and in an efficient way. Techniques drawn from computer vision and artificial intelligence aim to endow the system with the flexibility that humans seem to have when undertaking such a difficult task.

2 Description of the system

The present system evaluates the resulting solutions employing a performance measure that is widely used in the literature, known as the *packing density* (PD). This measure is defined as the ratio of the total

area of all the packed shapes to the area of their enclosing rectangle (1). The maximum value is 1 which implies that there is no waste of resource material.

$$\text{Packing Density} = \frac{\text{Area of Nested Parts}}{\text{Area of Enclosing Rectangle}} \quad (1)$$

The core algorithm of this system has some similarities with our algorithm in [2], where the case of the off-line packing problem was examined. In the present paper, modifications of this algorithm have been made, so that it can be applied effectively to on-line packing. A detailed description of the new system follows.

First of all, the sheet and the part that has to be placed each time are given as inputs to the system in the form of bitmap images. The placement of the parts follows a single-pass placement strategy. Two approaches of placing a part are considered before the system actually places it. These approaches are: *the Object-Based Approach* and *the Scene-Driven Approach*. Each of them suggests a potential placement of the part under consideration. The system carries out that placement of the part that gives the best packing density. In case both potential placements achieve the same packing density, the system favors the placement that fixes the part most bottom in the sheet.

2.1 The Object-Based Approach

The first approach examines four possible placements of the part. This is due to the four orientations that the part could take so that one side of its MER (Minimum Enclosing Rectangle) is aligned with a side of the sheet. Considering each one of these orientations, the system determines the most bottom-left position that the part can take without overlapping with the already nested parts and measures the packing density (1). For accomplishing the aforementioned procedure, the system utilizes operations of mathematical morphology [5]. As the layout and the part have both been stored in the system as images, let's denote the layout as the image set A and the part as the structuring element B . The operation of *erosion* is applied to the image set A by the structuring element B , denoted by $A \ominus B$ (2):

$$A \ominus B = \{x \in \mathbb{Z}^2 | B_x \subseteq A\} \quad (2)$$

where B_x denotes the translation of B so that its origin is located at x . By this operation, the set of all elements x of \mathbb{Z}^2 , such that B translated to x is contained in A , are identified [5]. The result of this operation is another image where all the feasible locations for the oriented part have been identified. The system keeps the most bottom-left placement and calculates the packing density. The Object-Based Approach has similarities with the method proposed by Whelan et al. [8].

¹ Department of Computing, Imperial College London, London SW7 2AZ, U.K., email: {alexandros.bouganis, m.shanahan}@imperial.ac.uk

2.2 The Scene-Driven Approach

The second approach searches for an unoccupied, by the already nested parts, region in the sheet, the shape of which matches well with the shape of the underlying part. This is carried out by temporarily approximating the shapes of the void regions in the layout and the parts by polygons [4], and employing the method of Arkin et al. [1] for measuring their resemblance. Let's consider two polygons A and B , as the approximated contours of a part and a void region respectively. Their resemblance can be measured by taking the minimal distance (according to the L_2 metric) between their turning functions $\Theta_A(s)$ and $\Theta_B(s)$ [1]. Thus, the similarity measure M of these two shapes is given by:

$$M = \min_{\theta_0 \in \mathbb{R}, t \in [0,1]} \left(\int_0^1 (\Theta_A(s+t) - \Theta_B(s) + \theta_0)^2 ds \right)^{\frac{1}{2}} \quad (3)$$

As proved in [1], the angle θ_0 that minimizes the distance between A, B is given by:

$$\theta_0 = \int_0^1 (\Theta_B(s) - \Theta_A(s)) ds \quad (4)$$

When the shape of a void region VR_i matches well with that of a part (i.e. M meets a predefined threshold), the system rotates the underlying part by the correspondent angle θ_0 and determines its potential location in the layout. Similarly to the Object-Based Approach, the potential location of the part is being found using the erosion operation. The void region VR_i and the oriented part comprise the image set and the structuring element respectively. Finally, the system measures the packing density (1) of this potential placement.

3 Performance Evaluation

The performance of any packing algorithm depends significantly on the parts' geometry, especially in the case of irregular packing. Thus, our packing method can only be evaluated properly when it is compared against other methods using benchmark problems from the literature. The proposed method is evaluated using nine benchmark problems, available from the EURO Special Interest Group on Cutting and Packing (ESICUP) website² and produced by Hopper [6]. The experiments conducted in this paper have been performed on a PC with a 3GHz Intel Pentium 4 processor and 1 GB RAM.

Aiming to test our method appropriately in on-line packing, a random generator is used and, for each benchmark problem, 20 experiments with random part orderings are examined. The results are illustrated in Table 1, which demonstrates, for each benchmark problem, the average value of packing density, the best packing density achieved among the 20 experiments, and the average execution time per experiment. As it can be seen, the system achieves solutions of high quality, requiring some milliseconds per placement of a part, fact that renders it well suited to on-line industrial applications.

To the authors' knowledge, no other on-line packing method of the literature has been applied to benchmark problems. However, we can consider as measure of comparison the performances of *off-line* methods that have been applied to the same benchmark problems (see [3], [6]). Although it is much harder for an autonomous system to achieve high quality solutions in on-line packing than it is in off-line, our algorithm proved to obtain solutions of comparable quality to the aforementioned off-line methods, in extremely shorter

times. Moreover, the structure of these methods does not allow them to be applied to on-line packing. It must be mentioned that our algorithm, in contrast to ([3], [6]), achieves these solutions without considering orientation constraints for the parts. Although their methods may find even better solutions without these constraints, their computational time would become prohibitive due to the large search space.

Table 1. Performance of our method on Hopper's benchmark problems. By Average PD and Best PD, we denote the Average Packing Density and the Best Packing Density achieved from our method among the 20 experiments for each benchmark problem, respectively. Moreover, the average time required per experiment, for each benchmark problem, is provided.

Benchmark problem	Number of parts	Average PD (Average time per experiment)	Best PD
poly1a	15	60.6% (4 sec)	71.1%
poly2a	30	64.3% (12 sec)	69.2%
poly3a	45	66.4% (23 sec)	68.7%
poly4a	60	67% (36 sec)	69.8%
poly5a	75	68.3% (51 sec)	70.4%
poly2b	30	65.5% (12 sec)	72.1%
poly3b	45	68.4% (23 sec)	71.9%
poly4b	60	68.2% (36 sec)	71.5%
poly5b	75	69.2% (56 sec)	73.4%

4 Conclusions

In this paper, an intelligent system for on-line packing of two-dimensional irregular shapes is presented. Computer vision and artificial intelligence techniques are utilized for endowing the system with the flexibility that an autonomous system needs to have while tackling such a difficult task. The proposed method is assessed on nine established benchmark problems and achieves high quality solutions in significantly short times.

REFERENCES

- [1] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell, 'An Efficiently Computable Metric for Comparing Polygonal Shapes', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**, 209–216, (1991).
- [2] A. Bouganis and M. Shanahan, 'A Vision Based Intelligent System for Packing 2-D Irregular Shapes', *IEEE Transactions on Automation Science and Engineering*, conditionally accepted.
- [3] E.K. Burke, R. Hellier, G. Kendall, and G. Whitwell, 'A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem', Accepted for publication in *Operations Research*, (to appear 2006).
- [4] D.H. Douglas and T.K. Peucker, 'Algorithms for the reduction of the number of points required to represent a digitised line or its caricature', *The Canadian Cartographer*, **10**, 112–122, (1973).
- [5] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, volume I, Addison-Wesley Publishing Company, 1992.
- [6] E. Hopper, *Two-dimensional Packing utilizing Evolutionary Algorithms and other Meta-Heuristic Methods*, Ph.D. dissertation, Cardiff University, 2000.
- [7] P. Kopardekar and A. Mital, 'Cutting stock problem: a heuristic solution based on operators intuitive strategies', *Int. J. Computer Integrated Manufacturing*, **12**, 371–382, (1999).
- [8] P.F. Whelan and B.G. Batchelor, 'Automated Packing Systems-A Systems Engineering Approach', *IEEE Transactions on Systems, Man and Cybernetics, Part A*, **26**, 533–544, (1996).

² <http://www.apdio.pt/sicup/>

Aliasing maps for robot global localization

Emanuele Frontoni and Primo Zingaretti¹

Abstract. In this paper we present a mobile robot localization system that integrates Monte-Carlo localization with an active action-selection approach based on an aliasing map. The main novelty of the approach is in the off-line evaluation of the perceptual aliasing of the environment and in the use of this knowledge to perform localization processes faster and better. Preliminary results show improved performances compared with the classic Monte-Carlo localization approach.

1. INTRODUCTION

Localization is one of the most basic abilities for a mobile robot. To localize, i.e., to be able to determine its own position in the environment, the robot needs some kind of representation of the environment.

Explicit (or geometric) representations are based either on maps of free spaces in the environment (using, for instance grid maps or polygonal-based representations) or on maps with locations of distinct observable objects (i.e., landmarks). Both methods rely on the assumption that geometric information (shape and position of obstacles, landmarks, etc.) can be extracted from robot's sensor readings. However, the transformation of sensor readings into geometric information is, in general, complex and prone to errors, increasing the difficulty of the localization problem [4].

In general, the localization model is implemented as a background process (i.e. localization is performed while the robot is involved in another task). However, when the robot is uncertain about its position it makes little sense to continue with the navigation task and it is more advisable to reduce the uncertainty in the robot position first, then continue with the navigation. Therefore, sometimes robot's actions have to be determined by localization criteria and not by navigation related ones. The question in such cases is how to determine the appropriateness of actions as far as localization is concerned.

In the context of mobile robot localization, actively controlling a robot is particularly beneficial when the environment possesses relatively few features that enable a robot to unambiguously determine its location. This is the case in many office environments. For example, corridors and offices often look alike for a mobile robot, which is often unable or very inefficient in determining robot's position, resulting in random motion or perpetual wall following.

Other approaches were presented to solve this kind of problem dealing in particular with the concept of entropy minimization [1, 3, 5].

We propose a mobile robot localization system that integrates Monte-Carlo localization with an active action-selection approach based on an aliasing map. The concept of aliasing maps is introduced in the next section, while results and discussion are reported in the final section.

2. ALIASING MAPS

The main idea of the proposed approach is to evaluate off-line the perceptual aliasing (i.e., different places that perceptually appear the same) of the environment and to use this knowledge to perform localization processes faster. In particular, we exploit this knowledge to bring the robot to a less aliased position in the environment, thus disambiguating robot belief and allowing a faster and more accurate localization.

The approach here proposed for off-line perceptual aliasing evaluation over a known map is general to every kind and number of range sensors (e.g., laser or sonar).

An aliasing map is defined by the function $\text{Aliasing}(x, y)$ that is a measure of the similarity of a certain point with other points of the map. In our approach we do not consider rotations to obtain a rotational invariant map: indeed measures used for aliasing evaluation covers 360 degrees around the robot.

The map is divided in areas reachable or not from the robot. A mesh is defined over the map of the environment and for each node of the mesh in the reachable areas a simulated sensor reading is performed. Sensor readings are range measurements without any particular need about noise modeling. For this reason a sensor measurement is divided into different steps. A sensor reading is performed for every node and its aliasing is measured as the number of sensor readings in different nodes that are in the same range step of the current reading. This approach is repeated for every sensor available in the robot, resulting in n_{Sensors} similarity values for every mesh node ($MeshNodes$ is the total number of mesh nodes). Results are stored in the matrix AliasingMatrix whose number of rows and columns is equal to the number of mesh nodes and different range sensors, respectively.

To obtain the map that represents the perceptual aliasing of every node of the mesh, the above similarities are fused together evaluating the following expression:

$$\prod_{j=1}^{n_{\text{Sensors}}} \frac{\text{AliasingMatrix}(k, j)}{\sum_{i=1}^{MeshNodes} \text{AliasingMatrix}(i, j)}, k = 1, \dots, MeshNodes$$

¹ Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione (DIIGA) - Università Politecnica delle Marche Ancona, Italy. E-mail: {frontoni, zinga}@diiga.univpm.it

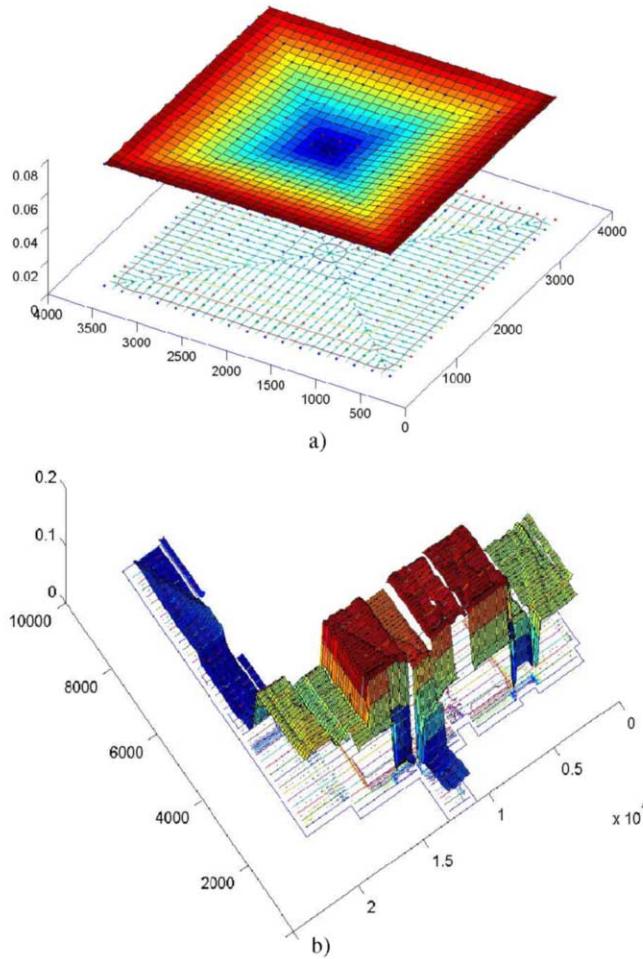


Fig. 1. a) The aliasing map of a squared environment where every point has a constant aliasing and the minimum of perceptual aliasing is in the square's center. b) A real aliasing map of an indoor environment of our department.

The mesh cells are then interpolated using a bi-cubic interpolation algorithm to obtain a fine grid aliasing map.

The map will be used in the algorithm to evaluate the direction that brings every particle to a minimum aliasing position in the environment (Fig.1).

The weighted sum of these optimal particle directions will be the optimal robot movement.

We assume that robot knows its initial orientation with a certain error. The direction of the best robot's action is given by the sum of all the minimum aliasing directions of each particle that allow to bring robot and particles to a minimum aliasing region of the environment.

The computational cost of the action selection is very low. Indeed only a differential evaluation for each particle is performed and this computation can be performed off-line and stored in a look up table.

3. RESULTS

Results have been obtained using a robot simulator in a high aliased environment. The used framework is a mixed Matlab – C++ mobile robotics platform that well suites to research and educational purposes. Further details about the framework can be found in [2].

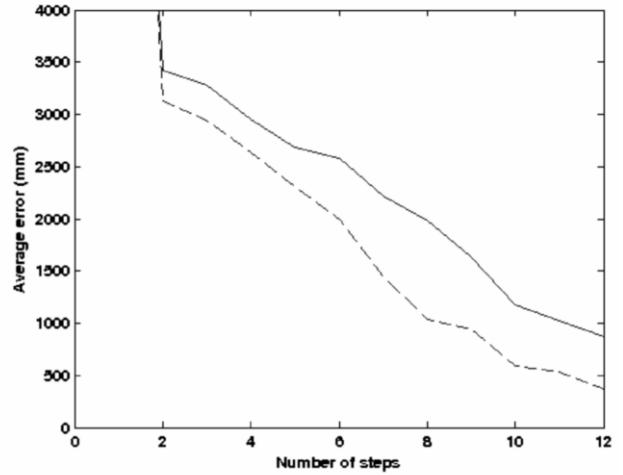


Fig. 2. Results of tests to compare classical particle filtering (continuous lines) approach and the proposed algorithm (dashed lines).

Fig.2 shows the better results obtained using the proposed approach. Results are obtained using 200 particles with a localization process that ends after 12 robot's steps. Due to the randomness of the MCL algorithm, results are the mean value of 10 localization processes starting from the same position.

4. CONCLUSIONS

The paper advocates a new, active approach to mobile robot localization. In active localization the robot controls its various effectors to most efficiently localize itself. In essence, actions are generated by maximizing the expected decrease of uncertainty, measured by perceptual aliasing.

The key result of the experimental comparison is that the efficiency of localization is increased when actions are selected minimizing global aliasing. In some cases, the action selection component enabled a robot to localize itself where the passive counterpart failed.

The main drawback of this approach is on its application to dynamic environments. Actually the off-line map aliasing evaluation does not guarantee a reconfiguration of algorithm parameters when dynamics events happen (i.e., people moving around the robot).

REFERENCES

- [1] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.
- [2] E. Frontoni, F. Mancini, A. Caponetti and P. Zingaretti, A framework for simulations and tests of mobile robotics tasks; Proc. of IEEE Mediterranean Conference on Control, Ancona, Italy, 2006.
- [3] E. Frontoni, P. Zingaretti, A vision based algorithm for active robot localization, Proc. of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, Helsinki, 347-352, 2005.
- [4] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, September 2005. ISBN 0-262-20162-3.
- [5] P. Zingaretti, E. Frontoni, Appearance-based localization in partially explored environments, *IEEE Robotics and Automation Magazine*, 13(1), 59-68, 2006.

On Interfacing with an Ubiquitous Robotic System

Donatella Guarino¹ and Alessandro Saffiotti²

Abstract. The emerging field of ubiquitous robotics presents new challenges for human-robot interface. In this note, we introduce the concept of a *common interface point* using an *expression-based semantics* as a way to address some of these challenges. We illustrate this concept in the framework of the PEIS-Ecology approach to ubiquitous robotics.

1 Introduction

There is a marked tendency toward the embedding of many ubiquitous, intelligent, networked robotic devices in our homes and offices. This tendency is witnessed by the growing interest in the fields of ambient intelligence and smart homes, as well as by the new emerging field of ubiquitous robotics [1, 3]. The development of these ubiquitous systems is often motivated by the desire to improve the quality of life of citizens in general, and of elderly people in particular. In this context, it is essential that this development is accompanied by the development of suitable *interfaces* that ensure the usability and acceptability of these systems.

The problem of interfacing with an ubiquitous system is different from, and more complex than, the conventional human-computer and human-robot interface problems. First, although the system consists of a collection of many inter-connected devices, the user should perceive it as one system, and interact with all the devices in it through *one and the same* interface. Second, the interaction between the user and (the different devices in) the system should be based on a *uniform model*, hiding the heterogeneity of the devices. Note that this interaction should be *two-way*, that is, the user should be able both to control the system and to get information from it.

In this note, we study how an ubiquitous robotic system can provide a human user with information which is unobtrusive, intuitive and relevant. We introduce two concepts that address the two challenges above: the concept of *common interface point*, which collects and summarizes the relevant information about the status of the system; and the concept of *expression-based semantics*, which provides a uniform semantics to talk about status information, and an intuitive way to convey this information to the user. In order to illustrate these concepts, we consider a specific concrete approach to ubiquitous robotics: the PEIS-Ecology approach.

2 The PEIS-Ecology Approach

The concept of PEIS-Ecology [3] puts together insights from the fields of autonomous robotics and ambient intelligence to generate a radically new approach to building assistive, personal, and service

robots. The main constituent of a PEIS-Ecology is a *physically embedded intelligent system*, or PEIS. This is any computerized system interacting with the environment through sensors and/or actuators and including some degree of “intelligence”. A PEIS can be as simple as a smart toaster and as complex as a humanoid robot. Notice that PEIS are *physically embedded systems*, not just software agents.

Individual PEIS in a PEIS-Ecology can cooperate by *linking functional components*: each PEIS can use functionalities from other PEIS in the ecology in order to compensate or to complement its own. The power of the PEIS-Ecology does not come from the individual power of its constituent PEIS, but it emerges from their ability to interact and cooperate. In a sense, the PEIS-Ecology approach redefines the very notion of a *robot* to encompass the entire environment.

Consider the example in figure 1. By itself, the simple autonomous cleaner can only use basic reactive cleaning strategies. If the home is equipped with an overhead tracking system, though, the robot can use it to know its own position in the house and hence use smarter cleaning strategies. Suppose then that the cleaner encounters a plant, and that the plant is equipped with a micro-PEIS (e.g., a mote) able to communicate its properties (size, wheel-support, humidity, temperature, etc) to the robot. Then, the robot can decide if it can push the plant away and clean under it.

3 The Common Interface Point

In a PEIS-Ecology, the single robot servant assumed in most literature on human-robot interface is replaced by a network of devices distributed all around the environment. Correspondingly, the notion of a single-robot interface should be replaced by a multitude of PEIS-specific interfaces, which are distributed all over the environment.

However, asking the user to interact with a large number of devices using a correspondingly large number of specialized and possibly different interfaces is clearly a bad idea. We therefore envision one single main *interface point* that integrates all the information from the different PEIS in the ecology, and dispatches all the com-



Figure 1. A simple PEIS-Ecology consisting of three PEIS.

¹ ICAR, National Council of Research. Viale delle Scienze, Palermo, Italy.
 Email: dony@pa.icar.cnr.it

² Center for Applied Autonomous Sensor Systems. University of Örebro, Sweden. Email: alessandro.saffiotti@aass.oru.se.

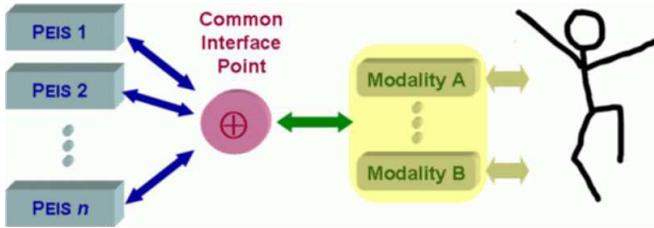


Figure 2. The common interface point for a PEIS-Ecology.

mands to them. This interface point may communicate with the user by a number of different modalities (e.g., speech, gestures, expressions), including remote ones (e.g., a mobile phone). The interface point, however, should be perceived as a single interface toward the whole ecology, as well as toward each individual PEIS. This concept is illustrated in figure 2.

In our work, we explore the realization of the common interface point in the system-user direction to convey status information. For the above schema to work, four problems must be solved: (1) how to encode status information from each PEIS, endowed with a *common semantics*; (2) how to *combine* status information coming from different PEIS, so as to reflect the status of the entire PEIS-Ecology (or parts of it); (3) how to *present* the status information to the user; and (4) how to *generate* this status information inside each PEIS. In the next section, we outline our solution to the first three problems; the fourth one is part of our current research.

4 Expression-Based Semantics

In many cases, a PEIS has an internal status from which we can distinguish “normal” and “abnormal” conditions. Different PEIS, however, have different ways to represent their status, and different measures to be used to detect a normal or abnormal condition. For instance, in the above scenario, the plant may need watering, or the cleaning robot may be running out of batteries: the relevant measures are the ground humidity or the battery voltage, respectively.

In order to provide a uniform way to treat these conditions we consider an abstract notion of “satisfaction”. Each PEIS represents its internal satisfaction status in a succinct way as an *expression*, ranging from “sad” to “happy”, and encoded by a real number in the $[0, 1]$ interval. These numbers are combined into a joint measure of the degree of satisfaction of the whole PEIS-Ecology, through some combination operator \oplus . Intermediate combination steps may be performed if the PEIS-Ecology is partitioned into sub-ecologies.

The satisfaction status is shown to the user in the form of a human understandable expression, which gives an immediate impression about the overall status of the PEIS-Ecology. Figure 3 shows some realizations of these expressions in our test implementation.

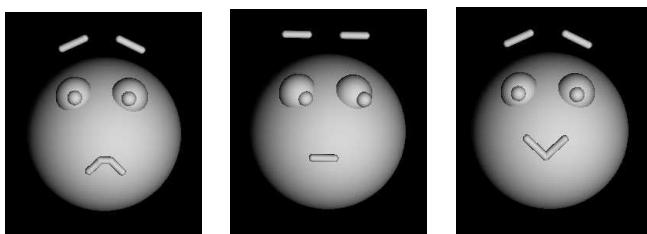


Figure 3. Visualization of the satisfaction status for three different values: 0.2 (very sad, left); 0.6 (rather worried, middle); 1.0 (quite happy, right).

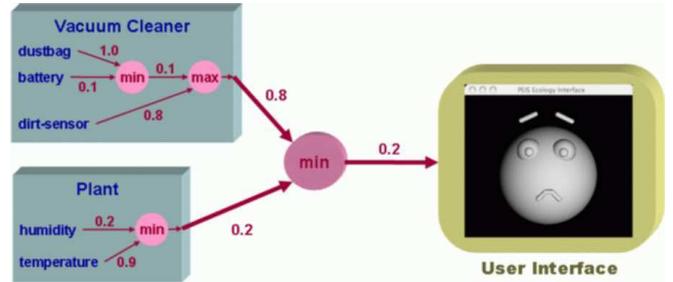


Figure 4. The overall satisfaction status of a simple PEIS-Ecology is computed, and it is conveyed to the user by a corresponding expression.

In general, the satisfaction state of each individual PEIS may depend on the combination of a number of different factors. For example, the state of the autonomous vacuum cleaner will be “happy” if the batteries are not empty and its dust-bag is not full. However, if the room is already clean, its state will be “happy” even if the batteries are empty, since no more cleaning is needed. This example shows that the behavior of the \oplus operator may need to be fairly complex, and that we may need different operators inside each PEIS, and between different subsets of PEIS.

To allow this, we combine satisfaction states using the mechanisms of fuzzy logic. Fuzzy logic has been given semantics in terms of desirability values [2]. These semantics are compatible with the expression semantics adopted here, by stipulating that the degree of satisfaction of a PEIS in a given state corresponds to the degree by which that state is *desirable* for that PEIS. Fuzzy logic provides mechanisms to write and evaluate logical combinations of $[0, 1]$ desirability values. For example, the following formula in (propositional) fuzzy logic could express the combined desirability for the vacuum cleaner example above:

$$\text{cleaner-happy} \Leftrightarrow \text{clean-floor} \vee (\text{full-battery} \wedge \text{empty-dustbag}) .$$

The degree of happiness of the cleaner PEIS, then, is obtained by computing the truth value of the above formula using the rules of fuzzy logic. Figure 4 shows an example of computation of the overall state of a simple two-PEIS ecology. In this example, the vacuum-cleaner is fine but the plant needs attention.

5 Next Steps

The study of expression-based semantics for ubiquitous systems has just started, and very many issues remain to be investigated. Among these: How can each PEIS generate its satisfaction value? How can we weight these values to take into account the user’s needs and preferences? How can we extend the expression-based semantics to other types of states, like danger or surprise? How can we convey “expressions” using other modalities, like sound or color? How can we help the user to pinpoint the cause of a “sad” status?

REFERENCES

- [1] K. Ohara, K. Ohba, B.K. Kim, T. Tanikawa, and S. Hirai, ‘System design for ubiquitous robotics with functions’, in *Proc of the 2nd Int Conf on Ubiquitous Robots and Ambient Intelligence*, (2005).
- [2] E.H. Ruspini, ‘On the semantics of fuzzy logic’, *Int. J. of Approximate Reasoning*, **5**, 45–88, (1991).
- [3] A. Saffiotti and M. Broxvall, ‘PEIS Ecologies: Ambient intelligence meets autonomous robotics’, in *Proc of the Int Conf on Smart Objects and Ambient Intelligence (sOc-EUSAJ)*, pp. 275–280, Grenoble, France, (2005). See <http://www.aass.oru.se/~peis/>.

Leaf classification using navigation-based skeletons

Georgios Sakellariou and Murray Shanahan¹

Abstract. In this paper, we present a leaf classification method based on skeletons produced by a navigation-inspired technique. The classification system comprises three separate stages. First, a *navigation-based* skeletonisation algorithm is used to gather low level structural and morphological information about the shape. Subsequently, the data is converted into a series of attributed graphs. Graphs of the same type are then compared using an approximate graph matcher, which identifies a degree of similarity between them. Each degree of similarity corresponds to a *dimension* in a *conceptual space*, as defined by Gärdenfors. We test the performance of our technique on a set of leaves belonging to three different species.

1 Introduction

While vision-based classification systems offer solutions to various problems in areas ranging from mobile robotics to medicine, less application-dependent methods of classification have not been as abundant. Our proposed system aims to overcome this issue, by following a modular object description and classification procedure. We apply our technique to the challenging problem of leaf classification, without using any expert knowledge with regard to the leaf types to be classified. The technique consists of the following stages. First, *navigation-based skeletonisation* [7] is performed on the object. The skeletons are then converted into *attributed graphs*. Finally, the graphs are compared by means of an *approximate graph matcher* [9]. The similarities between pairs of graphs act as *dimensions* in a *conceptual space* [3]. Distances between points (*knoxels*) in this conceptual space will be a measure of similarity between them. We claim that this approach enables the decoupling of the performance of the system from the application. In part, it owes its adaptability to the underlying skeletonisation method. Unlike other object description techniques [4], the exploratory nature of our navigation-based skeletonisation algorithm allows it to identify points of structural importance as well as morphological attributes on the fly, which facilitates later high-level processing. In addition, the method is more generic than others, in the sense that it can be adjusted to create graph representations containing a variety of different sorts of morphological information. The development of the algorithm took place with the aim of using the data it yields for *geometric representation*. This is achieved by means of attributed graphs, and the definition of a similarity metric to compare them. This method features two major advantages. First, the skeletonisation process is not continuous. It consists of hops from one skeletal point to another, which makes it easy to identify landmarks such as junctions. The data produced is therefore more easily processed by a high-level graph matching algorithm, such as those proposed in [9][12][6]. Second, unlike other systems [5], this technique allows for the collection of any kind of

morphological information about the shape in one pass. Finally, all topological and morphological attributes are extracted with no post-processing on the skeletons.

2 Conceptual Spaces for shape comparison

A *conceptual space* is a *metric* space in which entities are characterised by a number of quality dimensions [3]. In this respect, the goal of our classification system is to not just consider a single shape descriptor, but several of them. The comparison of each of those descriptors will correspond to a different *dimension* in our conceptual space. Conceptual spaces have already been employed in vision applications [2]. To each conceptual space corresponds a *similarity metric*. In this way, a degree of similarity can be determined when comparing *knoxels* (points in the conceptual space). In our system, we make use of three dimensions, all of which are measures of similarity between attributed graphs. The low-level information from a navigation-based skeletonisation algorithm is converted into attributed graph form. The skeletonisation algorithm that was used for this work draws on techniques originating in mobile robot navigation. The most attractive element of such an approach is modularity. Clearly, skeletonisation alone is not sufficient to successfully describe, and hence distinguish between, a multitude of shapes. This problem stems from the fact that the skeleton of an object merely encodes structure. However, objects of similar shape can produce very different skeletons. Conversely, objects whose shape differs substantially can produce very similar skeletons. One solution to the aforementioned problem, which is inherent to skeletonisation, is to enrich the produced skeleton with additional *morphological information*, thereby creating an attributed skeleton [1]. *Shock graphs* go one step further and use grammars of shape for description and shape matching purposes [10][11]. Attributed skeletons have been successfully used in a wide range of applications, including river segment classification [8].

3 Graph representation and matching

NavGraphs are attributed graph representations of navigation-based skeletons. Each node and each edge hold additional attributes for the purpose of describing shapes in a more distinctive way. Attributed graphs were our representation of choice due to their capacity to contain both topological and morphological information. In order to make the transition from skeletons to attributed skeletons, additional attributes must first be associated with every skeletal point. To achieve this, two separate steps have to be made. First, attributes that are appropriate to the application have to be chosen. Second, this information has to be represented in a suitable way for further processing and comparison purposes. As described in section 2, we make use of the outcome of graph matchings as dimensions of our

¹ Imperial College London, United Kingdom email: {georgios.sakellariou, m.shanahan}@imperial.ac.uk

conceptual space. The *Approximate Graph Matcher* [9] immediately seemed an appropriate choice. The original motivation for the development of the algorithm has been various biology-related applications. Two features of the algorithm render it most suitable for classification in the context of conceptual spaces. First, the matcher allows for attributed graphs to be compared. Second, the nature of its input facilitates comparisons and identifications of matches between many different graphs in one go. The graphs that need to be classified are encoded in a text file. The program then compares every graph to every other graph in the file, which effectively acts much like a database. Hence, even when comparing a large number of graphs, the process of classification is uncomplicated.

4 The leaf classification system

In order to evaluate the performance of our proposed approach, a series of comparisons was carried out. The system was tested on a set of leaves belonging to three different species. The leaf images were obtained from the database contained in the free version of the *Leaves Recognition* software by LightSpeed Communications. For the purposes of our experiments, we used ten samples per species. Each comparison involved three graph matchings, corresponding to the three dimensions. The similarity between two knoxels is defined as being inversely proportional to the vector distance of their corresponding points in the three-dimensional conceptual space. The experimental procedure consisted of a series of comparisons of leaves belonging to the same species, followed by a series of inter-species comparisons. Intra-species comparisons consisted of one specimen of a species being compared with all the other available specimens of the same species. Inter-species comparisons involved comparing a selected specimen of a species with all the specimens belonging to different species. The results show that there is a clear distinction between inter-species and intra-species comparisons. All three types of graph representations belonging to leaves of the same species have been determined to be more similar than those belonging to different species. In turn, the average dissimilarity between knoxels of inter-species and intra-species comparisons was significantly large. This facilitates the classification process.

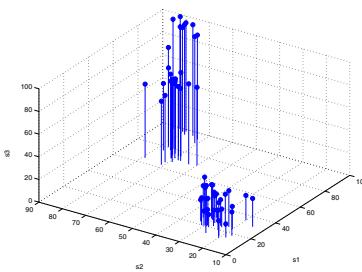


Figure 1. Large separation between intra-species and inter-species comparisons

The results can be viewed in terms of a three dimensional plot (Figure 1). The axes $s1$, $s2$ and $s3$ correspond to similarity using radius, rate of change of radius and angle respectively. We subsequently performed *Principal Components Analysis* on our set of data. The motivation for this was the fact that by determining the number of principal components needed to sufficiently describe the data, this analysis is an adequate method of proving the success of the system. We claim in this paper, that inter-species knoxels are clearly

distinguishable from intra-species knoxels, hence a reduction to one dimension would be a proof of this claim. Looking at the variance of the first principal component, one can draw conclusions concerning how much of the total information this component can describe. Indeed, almost all (97%) of the variance of all three principal components is contained in the first component. This proves that only one dimension is needed to include most information in the data set, implying that the polarisation between intra-species and inter-species knoxels is high.

5 Conclusions

In this paper we presented a leaf classification system based on a novel navigation-based skeletonisation algorithm. The extracted skeletons are transformed into multiple attributed graphs representations in a conceptual spaces inspired approach. This method proves advantageous in two major respects. First, by making attributed graph representation an integral part of the navigation-based skeleton, we eliminate the need for manipulation of the output of the skeletonisation algorithm. The extraction of descriptive attributes that are needed for matching and classification is done simultaneously with the skeletonisation process. Second, the fact that multiple graph representations of the same leaf are extracted, allows for a more generic and less application-dependent representation of the leaf.

REFERENCES

- [1] H. Blum, A Transformation for extracting new descriptors of shapes. Models for the Perception of Speech and Visual Form, Wathen-Dunn W (ed.), MIT Press, Cambridge, MA, 1967, pp. 362-380
- [2] A. Chella, M. Frixione, S. Gaglio (2001). Interpreting symbols on conceptualspaces: The case of dynamic scenarios. Proc. AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems, November 2nd-4th, 2001, North Falmouth, Massachusetts
- [3] P. Gardenfors, *Conceptual Spaces the geometry of thought*, MIT Press, 2000.
- [4] R. Meathrel and A. Galton, A Hierarchy of Boundary-based Shape Descriptors, in *Proceedings of IJCAI 2001*, Washington, U.S.A, August 2001, pp 1359-1364
- [5] Mokhtarian, F., and S. Abbasi, Matching Shapes with Self-Intersections: Application to Leaf Classification, *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 653-661, 2004.
- [6] N. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, California, 1980.
- [7] G. Sakellariou, M. Shanahan and B. Kuipers, Skeletonisation as mobile robot navigation, in *Proceeding of Towards Autonomous Robotic Systems (TAROS04)*, Essex, UK, 2004.
- [8] P. Santos, B. Bennett and G. Sakellariou, Supervaluation Semantics for an Inland Water feature Ontology, in *Proceedings of the 19th International Conference on Artificial Intelligence (IJCAI05)*, pp. 564-569, 2005
- [9] D. Shasha, J. T. L. Wang and R. Giugno. Algorithmics and Applications of Tree and Graph Searching, in *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002)*, Madison, Wisconsin, June 2002, pp. 39-52
- [10] K. Siddiqi and B. Kimia, Toward a Shock Grammar for Recognition, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1996
- [11] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker. Shock Graphs and Shape Matching. *International Journal of Computer Vision*, 30(1):1 24, 1999
- [12] J.R. Ullman, An algorithm for subgraph isomorphism, *Journal of the ACM*, 23(1):31-42, 1976.

Author Index

Agostini, A.	175	Bouguerra, A.	673
Ågotnes, T.	317	Boussemart, F.	113
Aknine, S.	180, 711	Boyer, A.	617
Alechina, N.	317, 322	Bratko, I.	148, 504
Alhajj, R.	494	Braun, L.M.M.	723
Almuhareb, A.	543	Breazeal, C.	3
Amalfi, M.	737	Bretier, P.	751
Amgoud, L.	235, 713, 747	Brewka, G.	xiii
Amigoni, F.	715	Briand, H.	357
Angelini, G.	819	Brinker, K.	489
Anger, C.	769	Brown, K.N.	701
Antoniades, A.	437	Buccafurri, F.	347
Artières, T.	797, 807	Burke, D.A.	701
Asker, M.	739	Byde, A.	300
Au, T.-C.	839	Cadoli, M.	68
Avouris, N.	41	Caminada, M.	743
Babik, M.	727	Caminiti, G.	347
Badea, L.	787	Cammisa, M.	548
Badra, F.	795	Carney, M.	791
Baiocetti, M.	575	Castagnos, S.	617
Balogh, Z.	727	Castelfranchi, C.	693
Balser, M.	835	Cerquides, J.	717
Barberá, S.H.	833	Cesta, A.	622, 845
Barry, J.	270	Chaib-draa, B.	729
Basili, R.	548, 568	Charitos, T.	745
Batenburg, K.J.	655	Charnley, J.	73
Beaufils, B.	185	Chatalic, Ph.	352
Becker, T.	612	Cheng, K.C.K.	78
Bell, J.	327	Chiarendini, M.	83
Benferhat, S.	332, 337, 741	Chong, Y.L.	108
Berkovsky, S.	789	Chopinaud, C.	205
Bertoli, P.	580	Cimatti, A.	580
Besnard, P.	763	Colton, S.	73
Bidyuk, B.	342	Coppola, B.	568
Bieler, H.	827	Coradeschi, S.	xiii
Bistarelli, S.	63, 705	Cortellessa, G.	622
Blaylock, N.	612	Cummins, R.	793
Bloch, I.	190	Cunningham, P.	627, 791
Boella, G.	195, 719, 721	Curatolo, D.	829
Bohte, S.M.	536	d'Aquin, M.	795
Bollegrala, D.	553	da Costa Pereira, C.	747
Bonatti, P.A.	200	Da Silva Neves, R.	11
Bonnefon, J.-F.	11, 16	Damiano, R.	31
Bonzon, E.	265	Dang, V.D.	210
Botti, V.	833	Dastani, M.	215, 220
Bouganis, A.	853	David, E.	285

David, J.	357	Gatial, E.	727
de Boer, V.	749	Gatti, N.	225
de Saint-Cyr, F.D.	235	Gaudou, B.	245
Dechter, R.	138, 342	Gebser, M.	769
Delany, S.J.	627	Gent, I.P.	98
Demetriades, I.	437	Gershman, A.	103
Demolombe, R.	751	Gerstenberger, C.	612
Denzinger, J.	260	Ghalla, M.	678
Di Gaspero, L.	83	Ghassem-Sani, G.	558
Di Giunta, F.	225	Ghédira, K.	275
Di Iorio, E.	531	Giordano, L.	757
Di Massa, V.	665	Giovannucci, A.	717
Di Noia, T.	230	Girgin, S.	494
Di Sciascio, E.	230	Giuglea, A.-M.	563
Ding, Y.	362	Giunchiglia, E.	377
Dipper, S.	827	Giunchiglia, F.	4, 382
Do, T.-M.-T.	797	GlioZZI, V.	757
Donini, F.M.	230	GilioZZO, A.M.	548
Druzzel, M.	482	Goldwasser, D.	789
Dubois, D.	11	Gori, M.	665, 819
Edelkamp, S.	841	Gottfried, B.	759
Edwards, M.	36	Gras, R.	357
Egly, U.	477	Grégoire, É.	387
Eiter, T.	367	Gressmann, J.	392
El Fallah Seghrouchni, A.	205	Groot, P.	835
Erdem, E.	367	Guarino, D.	857
Ernandes, M.	531, 819	Guillet, F.	357
Erny, J.	21	Halim, S.	703
Esposito, F.	765	Hamadi, Y.	108
Falappa, M.A.	402	Hameurlain, N.	713
Falcone, R.	693	Hasman, A.	723
Faltungs, B.	88	Heikkilä, T.	805
Farè, S.	715	Helmert, M.	585
Fargier, H.	16	Hemery, F.	113
Feili, H.	558	Henry, C.	823
Felfernig, A.	632	Herzig, A.	245, 397
Fermé, E.L.	402	Hess, M.	825
Fink, M.	367	Heymans, S.	462
Finzi, A.	753	Hluchy, L.	727
Fischer, F.	240	Hoffmann, A.	521
Forsell, N.	590	Hommersom, A.	835
Fratini, S.	622	Howard, C.	761
Frontoni, E.	855	Huang, S.Y.	305, 310
Fürnkranz, J.	489	Hüllermeier, E.	489
Gabaldon, A.	755	Hund, M.	660
Gadducci, F.	63	Hunter, A.	250
Galand, L.	93	Infantes, G.	678
Garcia, L.	372	Ingrand, F.	678
García-Pardo, J.Á.	833	Isak, K.	632

Ishizuka, M.	553	Lucas, P.	835
Jago, M.	322	Ludwig, B.	821
Janhunen, T.	392, 412, 769	Lukasiewicz, T.	753
Jansen, B.	26	Lunde, K.	647
Jefferson, C.	98	Lunde, R.	647
Jennings, N.R.	210, 285, 295, 300	Lundh, R.	683
Julián, V.	833	Maggini, M.	531
Jung, T.	499	Malec, J.	739
Junghanns, A.	637	Malfaz, M.	697
Junker, U.	118, 123	Mancini, T.	68
Kaci, S.	725	Mandl, S.	821
Kakas, A.	437	Maratea, M.	377
Kaljurand, K.	825	Marcos, M.	447
Karlsson, L.	673, 683	Marinescu, R.	138
Katreňák, B.	255	Marquis, P.	763
Katreňáková, J.	255	Mathieu, P.	185
Kern-Isberner, G.	402	Matsuo, Y.	553
Kidney, J.	260	Mattmüller, R.	585
Korthauer, A.	612	Maudes, J.M.	803
Koukam, A.	275	Mayer, W.	779
Kramer, O.	695	Mazure, B.	387
Kraus, S.	270	Meisels, A.	103
Kruijff-Korbayová, I.	612	Mercer, R.	392
Kuflík, T.	789	Mertsching, B.	660
Kurz, T.	642	Meyer, J.-J.Ch.	215, 220
La Poutré, J.A.	536	Meyer, T.	731
Laclavík, M.	727	Micaletto, D.	68
Lafrogne, S.	795	Miguel, I.	73, 98
Lagasquie-Schiex, M.-C.	265	Miksch, S.	447, 847
Landwerlin, L.	51	Milani, A.	575
Lang, J.	265, 763	Mileo, A.	781
Larrosa, J.	128	Monfardini, G.	665
Lau, H.C.	703	Moratz, R.	407
Laumonier, J.	729	Moro, G.	799
Lavagna, M.	715	Moschitti, A.	563, 568
Le Berre, D.	741	Mouaddib, A.-I.	843
Le Gloannec, S.	843	Moujahed, S.	275
Lecoutre, C.	113, 133	Možina, M.	504
Levesque, H.	5	Münker, B.	647
Lhomme, O.	123	Murray-Rust, D.	36
Lieber, J.	795	Myllymäki, P.	805
Lin, R.	270	Mylopoulos, J.	773
Lisi, F.A.	765	Nalbantov, G.I.	809
Lo Presti, K.	737	Napoli, A.	795
Lodi, S.	799	Nau, D.	839
Logan, B.	322	Nguyen, G.H.	352
Lombardo, V.	31	Nguyen, Q.-H.	88
Longin, D.	245	Nickles, M.	240, 245
Louis, V.	751	Nielsen, F.	509, 823

Nock, R.	509, 823	Prade, H.	11, 21, 337, 801
Norman, T.J.	280	Pralet, C.	427
Nunnari, F.	31	Preece, A.	280
O'Riordan, C.	793	Prosser, P.	707
O'Sullivan, B.	158	Provan, G.	783
Obradovic, Z.	526	Provetti, A.	737
Oddi, A.	622	Qin, Y.	526
Oikarinen, E.	412	Ragone, A.	230
Olivetti, N.	757	Ramos-Grijalvo, R.	833
Olmedilla, D.	200	Randell, D.	432
Oren, N.	280	Rasconi, R.	845
Öztürk, M.	417	Ray, O.	437
Paesold, M.	847	Rebollo, M.	833
Palmer, V.	733	Ricci, F.	607, 789
Palomares, A.	833	Rigutini, L.	531, 819
Pan, J.Z.	457	Rinaldi, F.	825
Papachristos, E.	41	Rintanen, J.	143, 600
Paroubek, P.	51	Rodríguez, J.J.	803
Pastor, J.	21	Rodríguez-Aguilar, J.A.	717
Patrizi, F.	68	Röger, G.	585
Payne, T.R.	285, 295	Rolland, A.	422
Peer, J.	200	Rollon, E.	128
Pekar, V.	516	Roos, T.	805
Perini, A.	xiii	Rosaci, D.	347
Perny, P.	93, 422	Rosenbrand, K.	447
Perrussel, L.	397, 731	Rossi, F.	705, 767
Peters, C.	46	Rousset, M.Ch.	352
Petit, E.	807	Russ, C.	632
Peyrard, N.	595	Sabbadin, R.	372, 590, 595
Pham, H.	773	Sadikov, A.	148
Pham, S.B.	521	Sadri, F.	442
Piette, C.	387	Saffiotti, A.	673, 683, 857
Pighin, D.	568	Sais, L.	113, 133
Pini, M.S.	705, 767	Sakama, C.	743
Pinkal, M.	612	Sakellariou, G.	859
Pino-Pérez, R.	190	Salichs, M.A.	697
Pistore, M.	580	Salvetti, F.	737
Pitz, M.	612	Sangiovanni, G.	715
Piunti, M.	693	Sartori, C.	799
Pizzo, A.	31	Sauro, L.	195
Poesio, M.	543	Scarselli, F.	665, 819
Poggioni, V.	575	Schaerf, A.	83
Polani, D.	499	Schaub, T.	392, 769
Polat, F.	494	Schehl, J.	612
Policella, N.	622, 845	Schiex, T.	427
Poller, P.	612	Schmitt, J.	835
Polo-Conde, C.	447	Schneider, G.	825
Poudade, J.	51	Sedki, K.	741
Pozzato, G.L.	757	Šefránek, J.	771

Seidl, M.	477	Tselios, N.	41
Senko, J.	367	Tsoukiàs, A.	417
Serrurier, M.	801	Unsworth, C.	707
Seyfang, A.	447, 847	Uzcátegui, C.	190
Shanahan, M.	853, 859	Vaillant, P.	823
Sharifi, M.	285	van den Herik, H.J.	723
Sharpanskykh, A.	290	van der Torre, L.	195, 719, 721, 725
Shehory, O.	180	van Dongen, M.R.C.	163
Shvaiko, P.	382	Van Nieuwenborgh, D.	462
Sicard, R.	807	van Riemsdijk, M.B.	220
Simonin, O.	275	van Someren, M.	749
Šiška, J.	771	Vanderlooy, S.	809, 811
Smaill, A.	36	Varzinczak, I.	397
Smirnov, E.N.	809, 811	Veale, T.	56
Smyth, B.	627	Venable, K.B.	705, 767
Soutchanski, M.	773	Venturini, A.	607
Sprinkhuizen-Kuyper, I.G.	809, 811	Verfaillie, G.	427
Stamatatos, E.	813	Vermeir, D.	462
Stamou, G.	457	Vidal, V.	133
Stede, M.	827	Vieu, L.	775
Stein, B.	695, 815, 829	Votruba, P.	847
Stein, S.	295	Wahlster, W.	6
Stéphan, I.	452	Wall, J.	695
Stergiou, K.	153	Walsh, T.	153, 168, 767
Stoffel, K.	642	Weng, P.	467
Stoilos, G.	457	Weydert, E.	725
Straccia, U.	457	Wielinga, B.J.	749
Stumptner, M.	761, 779	Wiesman, F.	723
Suriani, S.	575	Wilkenfeld, J.	270
Suriyawongkul, A.	827	Wilson, N.	472
Szathmary, L.	795	Witkowski, M.	432
Szymanek, R.	158	Wittenberg, J.	447
't Hoen, P.J.	536	Woltran, S.	477
Tabary, S.	133	Yap, R.H.C.	78, 703
Taillibert, P.	205	Yatskevich, M.	382
Tatar, M.	637	Yuen, D.C.K.	300
Tettamanzi, A.G.B.	747	Žabkar, J.	504
Thévenin, J.-M.	731	Zagorecki, A.	482
Thiele, S.	392	Zanuttini, B.	265
Tichy, R.	392	Zargayouna, M.	849
Tilivea, D.	787	Zhang, H.	305, 310
Toni, F.	442	Zhang, Y.	362
Traverso, P.	xiii	Zingaretti, P.	855
Treur, J.	290	Zivan, R.	103
Troquard, N.	775	zu Eissen, S.M.	815

This page intentionally left blank